uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería en Tecnologías de Telecomunicación
2018-2019

Trabajo Fin de Grado

"Diseño y desarrollo de un sistema de visión artificial programado en JavaScript para el navegador Web"

Ignacio Manzanares Romero

Tutor
Jesús Arias Fisteus

MADRID, JUNIO 2019

RESUMEN

El crecimiento de Internet y la inteligencia artificial han supuesto para el desarrollo tecnológico una nueva etapa. En ella, todas las actividades y trabajos especializados y repetitivos son realizadas por herramientas informáticas automatizadas.

Por eso, con el objetivo de aumentar la eficiencia y el desarrollo en el ámbito de la docencia se propone el presente proyecto: Emendábile, corrector de exámenes tipo test online.

El programa se basa en dos pilares esenciales: la visión artificial y el desarrollo Web. Fundamentado sobre esta base se propone una aplicación única, que en un entorno dinámico, interactivo, gratuito y libre de instalaciones, detecte y corrija a partir de una cámara digital exámenes tipo test realizados sobre un formato determinado.

Para ello, se han desarrollado los servicios de exposición, captura, detección y corrección en un entorno Web, haciendo uso del lenguaje de programación JavaScript.

Para la realización del proyecto, y debido a su similitud en el marco de la detección, se ha establecido como proyecto de referencia a Eyegrade, aplicación de escritorio con funcionalidad semejante. En procedimientos de Eyegrade se han basado los conceptos de los principales métodos de detección.

En el presente documento se expondrá el diseño, implementación y análisis del entorno realizados para la correcta realización de Emendábile.

Palabras clave: Computer Vision, Web services, Image recognition, JavaScript.

AGRADECIMIENTOS

A Javi, por su paciencia.

A Jesús, por su dedicación y atención a mi trabajo.

A Gonzalo, por sus consejos.

Muchas gracias.

ÍNDICE DE CONTENIDOS

RESUMEN **3**

1. INTRODUCCIÓN **13**

1.1 Motivación del proyecto ***13***

1.2 Objetivos ***14***

1.3 Introducción al resto de la memoria ***15***

2. PLAN DE TRABAJO Y PRESUPUESTO **17**

2.1 Plan de trabajo ***17***

2.2 Presupuesto ***19***

3. ESTADO DEL ARTE **20**

3.1 Desarrollo Web ***20***

3.1.1 Client-Side vs. Server-Side 20

3.1.1.1 Client-Side *21*

3.1.1.2 Server-Side *22*

3.1.1.3 Comparativa y elección. *24*

3.1.2 Client-Side Scripting 26

3.1.2.1 HTML *26*

3.1.2.2 CSS *29*

3.1.2.3 JavaScript *31*

3.2 Visión Artificial ***32***

3.2.1 Introducción a la Visión Artificial 32

3.2.2 Requisitos del sistema de Visión Artificial 33

3.2.3 Decisión 36

3.2.4 OpenCV 37

3.2.4.1 OpenCV.js *37*

3.3 Eyegrade ***38***

3.3.1 Eyegrade: Funcionamiento 38

3.3.2 Eyegrade: Diseño. 41

3.4 Soluciones similares ya existentes ***45***

4. ARQUITECTURA DEL SISTEMA **46**

5.1 Funcionamiento Emendábile ***46***

5. DISEÑO E IMPLEMENTACIÓN ***52***

5.1 Exposición ***52***

5.2 Captura ***56***

5.3 Detección ***57***

5.3.1 Preprocesado de la imagen 57

5.3.2 Detección de rectas 58

5.3.3 Detección de direcciones y ejes 59

5.3.4 Filtrado de rectas 61
5.3.5 Detección de celdas 62
5.3.6 Detección de respuestas 63
5.3.7 Detección de tipo de examen 67
5.3.8 Corrección de decisiones 68
5.3.9 Función process 69

6. CONCLUSIONES Y TRABAJOS FUTUROS 71
6.1 Conclusiones 71
6.2 Trabajos futuros 72

7. MARCO REGULADOR 73
7.1 Reglamento general de protección de datos 73
7.2 Licencias de herramientas utilizadas 73
7.3 Licencia propia de Emendábile 74

8. ENTORNO SOCIO-ECONÓMICO 75

LISTA DE ACRÓNIMOS 76

REFERENCIAS BIBLIOGRÁFICAS 77

ANEXOS 80
ANEXO A: ABSTRACT 80

ÍNDICE DE ILUSTRACIONES

Figura 1. Diagrama de Gantt. Planificación del proyecto.	18
Figura 2. El proceso de ejecución en el Client-Side.	21
Figura 3. El procedimiento en el Server-Side	22
Figura 4. Ejemplo de uso código HTML	28
Figura 5. Ejemplo de uso HTML.	28
Figura 6. Código de ejemplo lenguaje CSS.	30
Figura 7. Perfeccionamiento visual con CSS.	30
Figura 8. Ejecución secuencial del código web.	31
Figura 9. Ejemplo de decisión en Visión Artificial.	32
Figura 10. Ejemplo de nueva representación en Visión Artificial.	32
Figura 11. Transformación a binario (Adaptive Thresholding).]	34
Figura 12. Detección de rectas en imagen.	34
Figura 13. Dibujo de círculos sobre imagen vacía.	35
Figura 14. Formato de examen para la corrección en Eyegrade	39
Figura 15. Agregación de modelo o sesión en Eyegrade	39
Figura 16. Agregación de estudiantes para la posterior asignación de nota.	40
Figura 17. Formato rectas HoughLine.	42
Figura 18. Infobits. Detección del tipo de examen	43
Figura 19. Arquitectura principal de Emendábile.	46
Figura 20. Interfaz de inicio Emendábile.	47
Figura 21. Introducción de nuevo modelo. Paso 1.	48
Figura 22. Introducción de nuevo modelo. Paso 2.	49
Figura 23. Muestra de examen corregido.	50
Figura 24. Rectificación de detección.	51

Figura 25. Estructura básica de la interfaz gráfica.	52
Figura 26. Transformación de imagen a escala de grises	57
Figura 27. Transformación binaria adaptativa e inversa.	58
Figura 28. Detección de líneas de la imagen.	59
Figura 29. Filtrado de direcciones	60
Figura 30. Filtrado de rectas en función de su separación.	61
Figura 31. Código para la comprobación del número de líneas.	62
Figura 32. Puntos hallados por la intersección de las líneas	63
Figura 33. Creación de matriz de celdas.	64
Figura 34. Casilla marcada	65
Figura 35. Máscara de casilla marcada	65
Figura 36. Resultado de la multiplicación. Casilla marcada.	65
Figura 37. Casilla no marcada	66
Figura 38. Máscara de casilla no marcada	66
Figura 39. Resultado de la multiplicación. Casilla no marcada.	66
Figura 40. Ejemplo de matriz de decisiones.	67
Figura 41. Infobit positivo en imagen preprocesada.	67
Figura 42. Máscara circular de infobit positivo.	68
Figura 43. Resultado de la multiplicación. Infobit positivo detectado	68
Figura 44. Código de la función *correction,* para la asignación de nota.	69

ÍNDICE DE TABLAS

Tabla 1. Presupuesto total del proyecto.	19
Tabla 2. Comparación de arquitecturas Client-Side y Server-Side.	24

1. INTRODUCCIÓN

En este documento se pretenderá exponer, justificar y aclarar la investigación y desarrollo realizados para la implementación de Emendábile: un servicio web capacitado para la detección y corrección de exámenes tipo test mediante el uso de una cámara digital.

1.1 Motivación del proyecto

El procesamiento máquina y la Inteligencia Artificial son técnicas innovadoras y muy eficientes en la realización de actividades automáticas y repetitivas. Es por esto por lo que en los últimos años toda actividad con estas características está siendo sustituida por un procesamiento informático que aumenta cualitativamente su eficiencia.

Estas modificaciones y mejoras tecnológicas se han aplicado absolutamente en todos los campos posibles. Y precisamente la educación no es un ámbito exento.
Así, en este proyecto también se usará esta amplificación y perfeccionamiento tecnológico con el objetivo de mejorar y facilitar la docencia y enseñanza en cualquier entorno.

La metodología de examinación tipo test se caracteriza por su evaluación contextual y no argumental de las respuestas. Esta corrección precisamente forma parte del conjunto de actividades automáticas mencionadas anteriormente, fácilmente implementables en programas y funciones informáticas.
Por lo que será la informatización de este ámbito particular de la docencia el objetivo sobre el que se fundamente el proyecto. Es decir, en este proyecto se desarrollará una herramienta capaz de automatizar e informatizar la corrección de exámenes tipo test.

Sin embargo, ya existen herramientas que cumplen esta labor (*Sección 3.4*). Por eso, la diferenciación del servicio se centrará básicamente en su implementación en un entorno Web. De este modo se le brindan al usuario una serie de ventajas que de otra manera no serían viables:

- *Acceso desde cualquier plataforma*. El navegador web será el encargado de la compilación y ejecución del código, luego es irrelevante el sistema operativo del que disponga el usuario (Linux, Windows, IOS, Android, etc.), siempre y cuando permita acceder al servidor desde un navegador compatible.
  Esto además aporta una ventaja adicional en cuanto a flexibilidad en el uso, ya que la herramienta es compatible con cualquier dispositivo con navegador Web (Ej: PC, Smartphone, Tablet, ...)

- *Sin necesidad de instalación.* El uso de la aplicación se simplifica al no requerir una instalación inicial del producto, sino exclusivamente un acceso instantáneo al servidor.

- *Actualizaciones automáticas.* Por supuesto, todas las actualizaciones y mejoras se añadirían automáticamente desde el lado del servidor. Ya no serían necesarias las actualizaciones manuales del producto por parte del usuario, y éste accederá siempre a la última versión disponible.

Para el desarrollo global del proyecto se tomará como referencia el programa *Eyegrade* [1], que sólo está disponible como aplicación de escritorio, para desarrollar una implementación Web compatible con él.

Eyegrade, como se explicará con detalle más adelante, es un programa de código abierto que permite la creación de exámenes de múltiples opciones, su corrección mediante una webcam y la exportación de los resultados.

Teniendo en cuenta esta referencia, el proyecto a realizar se centrará en implementar las funcionalidades básicas de dicho programa en un entorno web.

1.2 Objetivos

El objetivo principal de este proyecto consistirá en la realización de un programa de corrección de exámenes tipo test con un formato determinado mediante el uso de visión artificial en un entorno web.
Además, se buscará la compatibilidad con Eyegrade, programa referencial de este proyecto.

El programa a desarrollar deberá implementar las siguientes funciones:

1. Inserción del modelo de examen en el programa. La aplicación permitirá al usuario introducir en un formato preestablecido el modelo de examen: cantidad de preguntas, cantidad de respuestas, puntaje, y las soluciones del test a corregir. De este modo la aplicación conocerá qué atributos debe detectar y cuál de las respuestas es correcta y cuál errónea.

2. Detección de exámenes. En el entorno del navegador web, el programa será capaz de acceder a la información de la webcam, detectar el área de soluciones, el tipo del examen y las respuestas marcadas en las casillas.

3. Muestra de corrección. El programa mostrará al usuario los resultados de la detección: las celdas, las respuestas marcadas con su calificación, nota resultante, etc.

4. Corrección de errores. El programa permitirá al usuario corregir los posibles fallos cometidos en la detección. Por ejemplo marcar como seleccionada una celda que el programa ha detectado como vacía.

Adicionalmente, se buscará que el proyecto cuente con las siguientes características:

1. Interfaz de usuario intuitiva, presentada de una forma sencilla y atractiva, que permita al usuario trabajar con las funciones enumeradas anteriormente

2. Programa libre, distribuido mediante licencia de código abierto.

1.3 Introducción al resto de la memoria

Este documento se organizará según las siguientes secciones:

1. *Introducción. Motivaciones y objetivos.*
   Presentación al documento y exposición de las motivaciones y objetivos que han determinado el trabajo realizado.

2. *Plan de trabajo y presupuesto.*
   Exposición de los recursos económicos y temporales necesarios para la correcta implementación y resolución del proyecto. Para ello se elaborarán las referencias visuales necesarias: Diagrama de Gantt y Tabla de Presupuestos.

3. *Estado del arte.*
   Análisis del entorno de desarrollo del proyecto: herramientas utilizadas, modelos de referencia, estado del mercado y posibles alternativas.

4. *Arquitectura del sistema.*
   Análisis global del diseño estructural de Emendábile, exponiendo su funcionamiento y el conjunto de módulos que forman su desarrollo.

5. *Diseño e implementación.*
   Presentación y desarrollo de cada una de los procesos principales del mecanismo de ejecución; su funcionamiento, desarrollo y justificación.
   Exhibición a bajo nivel de cada uno de los módulos del sistema, y de sus comunicaciones y relaciones entre sí.

6. *Conclusiones y trabajos futuros.*
   Estudio del proyecto realizado. Cumplimiento de los objetivos planteados y principales dificultades salvadas en el desarrollo del programa.
   Planteamiento de posibles mejoras a implementar en el programa en futuros hilos de trabajo.

7. *Marco regulador.*
   Análisis del entorno legislativo del programa: qué aspectos del marco regulador se han tenido en cuenta en la realización del proyecto, y cómo éste ha podido afectar al estado final del programa.

8. *Entorno socio-económico.*
   Estudio del impacto social y económico del proyecto realizado.

9. *Lista de acrónimos.*
   Enumeración de los acrónimos y abreviaturas empleados en el documento, con su significado y desarrollo.

10. *Referencias bibliográficas.*
    Conjunto organizado de las referencias externas usadas para la elaboración del documento.

2. PLAN DE TRABAJO Y PRESUPUESTO

2.1 Plan de trabajo

Para la realización del proyecto, será conveniente realizar una planificación global de las tareas a realizar, evaluando su importancia y aproximando su coste temporal.

Para una mayor visualización se realizará un diagrama de Gantt (**Figura 1**), herramienta para la representación cronológica de las tareas. Las tareas principales serán las siguientes:

1. ***Estudio y análisis del entorno***
   - 1.1. Estudio y análisis de Eyegrade
   - 1.2. Estudio y práctica de lenguaje JavaScript
   - 1.3. Estudio y Análisis de posibles librerías de Visión artificial
2. ***OpenCV.js***
   - 2.1. Estudio y Análisis de OpenCV.js
   - 2.2. Creación de entorno de construcción OpenCV.js
   - 2.3. Implementación de pruebas de las funciones principales
3. ***Preprocesado de la imagen.***
4. ***Detección de tabla de respuestas.***
   - 4.1. Detección de líneas
   - 4.2. Detección de direcciones
   - 4.3. Filtrado de líneas
   - 4.4. Detección de celdas
5. ***Detección de cruces.***
6. ***Detección de imágenes válidas del conjunto de vídeo.***
   - 6.1. Comunicación con el programa del contenido multimedia
7. ***Detección del tipo de examen en función de los infobits.***
8. ***Corrección y asignación de nota.***
9. ***Revisión y rectificación de la corrección.***
10. ***Introducción de nuevo modelo de examen.***
    - 10.1. Formulario de inserción
    - 10.2. Asimilación de soluciones
11. ***Elaboración de la memoria***
12. ***Depuración de código***
13. ***Desarrollo visual de la aplicación.***

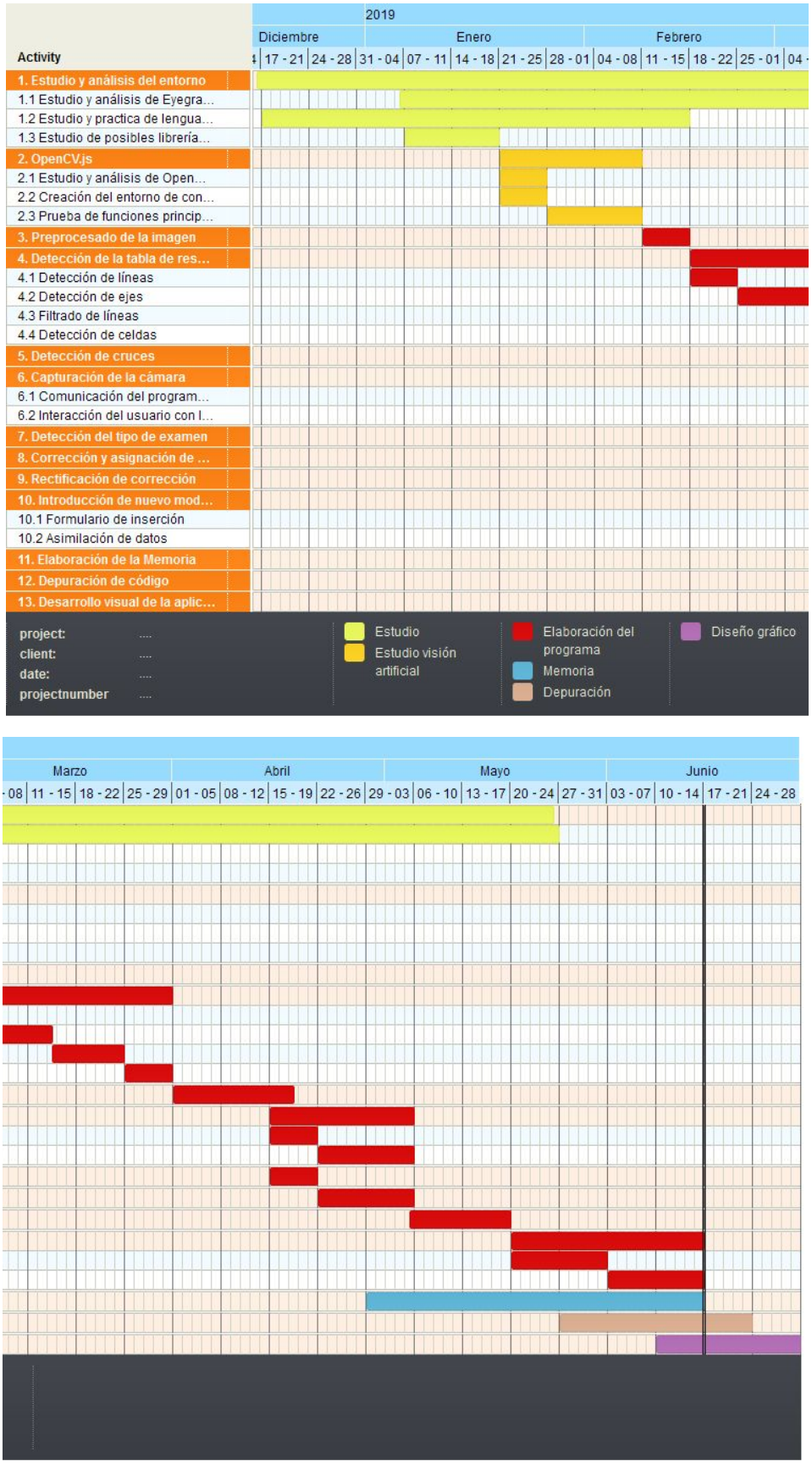

Figura 1. Diagrama de Gantt. Planificación del proyecto.

2.2 Presupuesto

El coste total aproximado para la realización del proyecto será la suma de los siguientes:

- Costes de material:
    - Ordenador para la programación e implementación → 1.000 €
        - Teniendo en cuenta un tiempo de desarrollo de 8 meses, y una vida útil del ordenador de 4 años. El coste amortizado para el proyecto será de **167 €**
    - WebCam → [Despreciable]
- Coste de personal:
    - 1 Desarrollador, con un trabajo de 300 a 350 horas, y coste de 15€ por hora de trabajo (sueldo bruto). → **4.875 €**
- Costes indirectos (Instalaciones, gastos de mantenimiento, etc.): Aproximadamente supone un 20% de los gastos anteriores → **1008,4 €**
- Impuesto sobre el Valor Añadido (IVA): 21% de los costes anteriores → **1.270,58 €**

TOTAL: 7.320,98 €

CONCEPTO	COSTE EN EL PROYECTO
MATERIAL	167 €
PERSONAL	4.875 €
COSTES INDIRECTOS	1008,4 €
IVA	1.270,58 €
TOTAL	**7.320,98 €**

Tabla 1. Presupuesto total del proyecto.

3. ESTADO DEL ARTE

Este capítulo se centrará en el entorno en el que se ha desarrollado el proyecto: las alternativas existentes al programa a desarrollar, las herramientas usadas, los referentes escogidos y las bases sobre las que se ha decidido desarrollar la aplicación frente a todas las alternativas posibles.

Para ello se analizarán cada uno de los dos aspectos principales del proyecto sobre los que se despliegan múltiples posibilidades de implementación: el desarrollo Web de la aplicación y la Visión Artificial. Dentro de cada uno de estos ámbitos se mostrará una panorámica de su contexto de desarrollo, exponiendo el conjunto de requisitos que debe cumplir, y se realizará una comparativa de rendimiento y adecuación de cada una de las alternativas posibles.
Por último, dentro de cada sección se expondrá la solución final escogida con los argumentos que justifican su elección.

Tras haber decidido el conjunto de herramientas y estructuras sobre las que se edificará el proyecto, se realizará un análisis de Eyegrade, como referente del proyecto, exponiendo su estructura, fundamento y funciones principales.

Finalmente se realizará un estudio de la competencia evaluando soluciones similares ya existentes, y comparando con el producto realizado, detallando la diferenciabilidad de Emendábile.

3.1 Desarrollo Web

Como ya se ha expuesto, el objetivo principal y cualidad diferencial de este proyecto es la implantación online de un programa de corrección basado en Eyegrade. Por ello, el desarrollo Web va a ser la base principal sobre la que se desarrollarán los requisitos del proyecto.
La intención para esta sección es que se consiga un entorno de ejecución del programa fluido, interactivo, sin instalaciones ni actualizaciones y multiplataforma.
Para ello será requisito indispensable la utilización de un servidor Web, en el que se implantará el programa y será accesible desde cualquier posible plataforma del cliente.

3.1.1 Client-Side vs. Server-Side

Existen dos posibles arquitecturas que permiten cumplir los requisitos de la implementación: Client-Side y Server-Side.
La mayoría de aplicaciones complejas hacen uso de ambas estructuras, dando más importancia a una o a otra. A continuación se explicará a grandes rasgos el funcionamiento de cada una.

3.1.1.1 Client-Side

Esta arquitectura se basa en la ejecución del código en el cliente. Por cliente se entiende la aplicación que realiza la petición al servidor, normalmente un navegador web. Un ejemplo de uso de esta arquitectura es el procesamiento de la interfaz gráfica de páginas como Netflix, que se ejecuta en cada dispositivo de forma distinta, adecuándose a sus características.

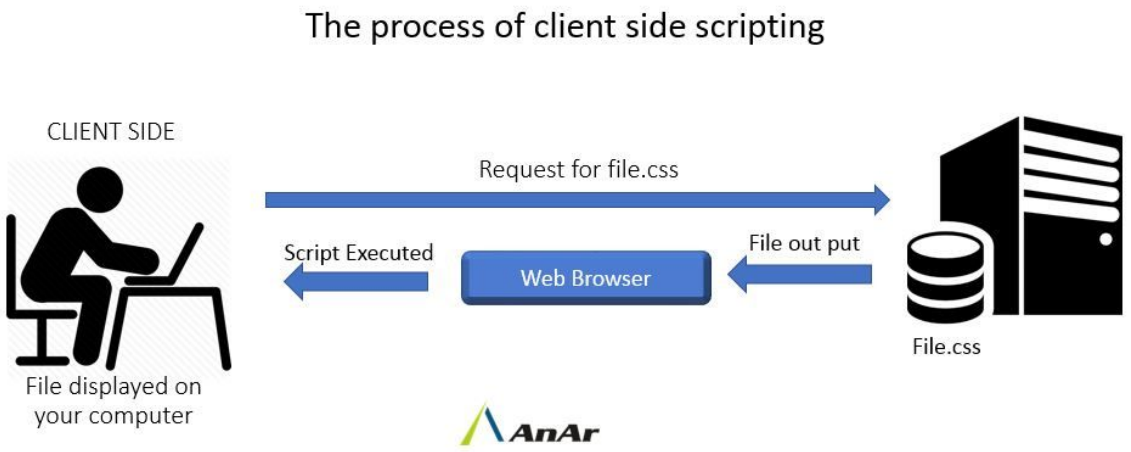

Figura 2. El proceso de ejecución en el Client-Side [2]

Como se observa en la **Figura 2** y en [2], el procedimiento de la ejecución del código es el siguiente:

1. El cliente realiza una petición HTTP al servidor con la solicitud del servicio.
2. El servidor comprueba que el cliente es compatible con el servicio a prestar y le responde con el código de ejecución dinámico del programa o *script*.
3. El cliente ejecuta el script suministrado y ofrece al usuario la información resultante.

Una aplicación se desarrolla desde el lado del cliente cuando se requiere de su información privada accesible exclusivamente por él o cuando el servidor no tiene capacidad suficiente para satisfacer las necesidades de todos sus clientes.

Las principales ventajas e inconvenientes en general y para nuestro proyecto, según [3] y [4], son las siguientes:

Ventajas Client-Side:

- Se reduce al máximo la comunicación entre cliente y servidor. Esto reduce tiempo de ejecución del programa y la saturación de la red.
- Se reduce la actividad del servidor, que ya no tiene que ejecutar simultáneamente el servicio a todos los clientes conectados, sino que su función es exclusivamente el suministro del código de los programas a ejecutar por el cliente.

- Aumento de privacidad y seguridad de datos personales. Ya que los datos introducidos no son procesados por el servidor ni transmitidos a través de la red, sino que son manipulados a nivel local exclusivamente por el navegador.
- Aumento de la velocidad de reacción del programa frente a errores o indicaciones, ya que no es necesaria la comunicación con el servidor para verificar cada interacción condicionante del usuario.

Desventajas Client-Side:

- Mayor amenaza a la aplicación. El cliente puede depurar el código o interrumpir su procesamiento natural. Esto supone una amenaza en el control del servidor y en el funcionamiento global de la aplicación.
- No todos los navegadores web son compatibles con el código proporcionado por el servidor.
- Mayor tiempo de diseño del programa, para satisfacer los requisitos de cada plataforma en su lectura del programa.
- Posible desactualización de contenido en la ejecución del programa, ya que la ejecución no está en comunicación directa con las bases y centros de datos.

Las principales herramientas o lenguajes utilizados para el desarrollo en el lado del cliente son HTML, CSS y JavaScript [1] , todas ellas diseñadas para este tipo de arquitectura.

3.1.1.2 Server-Side

Del mismo modo que la arquitectura Client-Side se centra en el entorno del cliente como marco de ejecución del programa, ahora nos encontramos, análogamente, con la arquitectura Server-Side.
Tal y como se explica en [4] y [5], esta estructura centra el medio de ejecución principal en el servidor online. El procedimiento general de esta metodología es el siguiente:

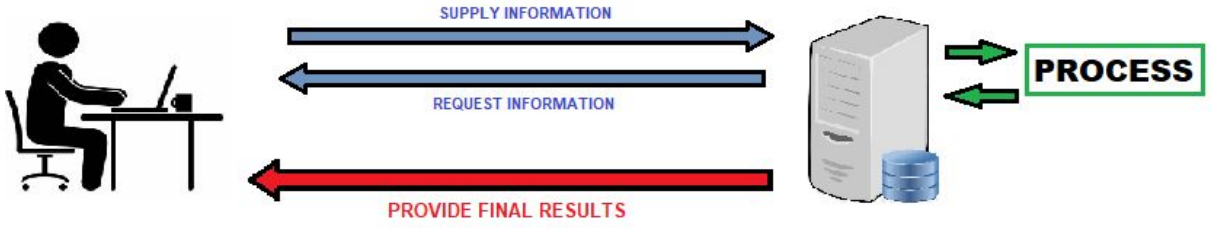

Figura 3. El procedimiento en el Server-Side

1. El cliente realiza la petición al servidor y se establece la conexión HTTP.
2. El servidor solicita (si no la tiene) la información necesaria para la ejecución del programa.
3. El Cliente otorga al Servidor la información necesaria.

[1] Recientemente es posible desarrollar código en otros lenguajes, como C o Rust, y compilarlo a código WebAssembly, formato del que hablaremos posteriormente, compatible con las versiones más recientes de los principales navegadores.

4. Se repiten paso 2 y 3 hasta la finalización del proceso.
5. El servidor entrega al Cliente el resultado del procesamiento.

Una aplicación se desarrolla en el lado del servidor cuando se quiere mantener un nivel de seguridad, se puede costear una ejecución multi-proceso más trabajosa y la saturación de la red no va a ser un problema.

Las principales ventajas e inconvenientes de la metodología Server-Side son las siguientes:

Ventajas Server-Side:

- Comunicación directa con bases de datos. Esto mejora el funcionamiento del programa por su continua actualización de contenidos.
- Posible mejora de tiempo de cómputo. Si la capacidad de los servidores es muy elevada, estos podrán ofrecer mejor prestaciones que los dispositivos de uso del cliente.
- Mayor privacidad en el código del programa, ya que éste no es accesible para el cliente. Esto implica una mayor seguridad ante ataques al código.
- La necesidad de compatibilidad del lenguaje es únicamente con el servidor, por lo que la implementación resultará más sencilla.

Desventajas Server-Side:

- Mucho mayor coste de infraestructuras e instalaciones. Ya que se debe asegurar la capacidad para servir a múltiples clientes simultáneos, es decir, es una ejecución muy costosa.
- La saturación de la red y aumento del ancho de banda utilizado. Esto incrementa los costes y propicia una ralentización del servicio en casos de aumento de clientes.
- Disminuye la dinamicidad de los procesos, al requerir la comunicación directa con el servidor para cada interacción.

Las herramientas que más se utilizan para desarrollar estructuras del lado del servidor son: PHP, Java, el entorno .NET, Python, JavaScript[2] y Ruby.

[2] Aunque sea un lenguaje orientado principalmente al lado del cliente, herramientas como Node.js [6] permiten y acomodan su uso para arquitecturas de Server-Side.

3.1.1.3 Comparativa y elección.

Una vez realizado el análisis individual de cada estructura, se mostrará una comparativa de sus principales características:

CLIENT-SIDE	SERVER-SIDE
Coste de servidores e infraestructuras **MENOR**	**Coste** de infraestructuras **ELEVADO**
Sin acceso directo a bases de datos. Posible **información desactualizada**	Conexión continua con centros de datos. **Información actualizada**
Mayor **dinamicidad**, no requiere conexión continua con el servidor	El acceso constante al servidor provoca **ralentización** y saturación.
MENOR seguridad, el código es accesible para cualquier usuario	**MAYOR seguridad**, el cliente sólo obtiene el resultado del código ejecutado.
La calidad de la **ejecución**, **dependiente** del dispositivo de uso **del cliente**	La calidad de la **ejecución**, **dependiente** exclusivamente de los recursos **de la compañía**.
Lenguajes principales: **JavaScript**, **HTML** y **CSS**	Lenguajes principales: **PHP**, **Java**, **ASP**, **Python** y **Ruby**

Tabla 2. Comparación de arquitecturas Client-Side y Server-Side.

Por sencillez y viabilidad, el desarrollo del proyecto se desarrollará exclusivamente en una de las dos arquitecturas analizadas. Para la elección apropiada de la estructura que se ha de utilizar se expondrán los requisitos de desarrollo Web de nuestro sistema:

- *Requisito 1.* Compatibilidad con Eyegrade, modelo de implementación. Es decir, se buscará que la solución resultante sea semejante en funcionamiento, diseño y uso a Eyegrade.

- *Requisito 2.* Bajo coste. Por el contexto de creación del proyecto es preferible una estructura que suponga el menor coste económico posible de implementación y funcionamiento.

- *Requisito 3.* Dinamismo. Al tratarse de una aplicación de visión artificial en directo, es decir, que muestra dinámicamente el vídeo que se está grabando, se pretenderá minimizar el tiempo entre captura y muestra de cada imagen.

- *Requisito 4.* Seguridad de datos personales. El carácter privado de los datos de los alumnos requiere de una protección especial frente a ataques de la información personal.

Analizaremos ahora la capacidad de adecuación de cada una de las dos estructuras vistas a nuestro proyecto.

Client-Side en Emendábile

Según las características expuestas de este modelo, si el programa fuese desarrollado para el lado del cliente se cumplirían la mayoría de los requisitos del proyecto (*Requisitos 2, 3 y 4*), pero se ha de estudiar la posibilidad de satisfacer el *Requisito 1,* es decir, compatibilizar esta estructura con Eyegrade.

A grandes rasgos (se examinará posteriormente), Eyegrade basa su funcionamiento en la utilización de la librería OpenCV para la visión artificial, utilizando como fundamento el lenguaje de programación Python.
Para diseñar el proyecto desde el lado del cliente, será necesaria la programación en JavaScript, luego si escogemos esta arquitectura habremos de rediseñar todas las funciones de Eyegrade en un nuevo lenguaje y entorno.

Server-Side en Emendábile

Acorde a las cualidades de esta estructura, y comparándolas con los requisitos necesarios para la realización del proyecto, se halla que:

El *Requisito 1* de fácil cumplimiento, ya que al estar Eyegrade diseñado para Python, y siendo éste un lenguaje compatible con la estructura Server-Side, a priori bastaría con su implementación casi directa en los servidores.
Sin embargo, ninguno de los otros tres requisitos se pueden cumplir sin dificultades excesivas. Especialmente el segundo, ya que el coste de las instalaciones necesarias siguiendo esta estructura sería demasiado.

Por estas incompatibilidades entre los requisitos del proyecto y la metodología Server-Side, y por el potencial de la técnica de la ejecución en el cliente, se tomará la decisión de implementar el proyecto en un entorno de Client-Side.

3.1.2 Client-Side Scripting

Una vez decidida la estructura principal del proyecto, se han de introducir las herramientas fundamentales sobre las que se edificará la implementación. Al tratarse de un entorno Web con ejecución en el lado del cliente se utilizarán las siguientes herramientas:

3.1.2.1 HTML

HTML es un lenguaje de programación web basado en la estructuración y definición de contenido en páginas web. Es un estándar a cargo del *Consorcio WWW* [3].
Se trata de un lenguaje orientado a la categorización funcional por etiquetas [8]. Es decir, la estructura y funcionamiento del código se basa en el correcto uso de etiquetas o *'Markups'* sobre los elementos.

Este etiquetado debe cumplir unos requisitos fundamentales:

- Correcta disposición de los elementos con un anidado adecuado.
- Apertura y cierre de etiquetas coherente.
- Escritura de etiquetas en minúsculas.
- Inserción de valores entre comillas.

El conjunto de etiquetas [9] se puede catalogar en:

1. *Metadatos del documento:*
   Etiquetas referenciales del documento. Contextualizan el código y aportan la información de base necesaria para el funcionamiento. Las etiquetas principales son:

   <head> (referencias generales: enlaces a, definiciones, scripts, hojas de estilo, ...); **<title>** (título del documento); **<style>** (Escritura CSS en línea); etc.

2. *Secciones:*
   Definen el esqueleto principal del documento. Su estructura jerárquica, importancia de contenidos y división principal.

   <body> (Cataloga el contenido principal del documento HTML. Sólo puede haber un elemento body); **<h1>, <h2>, <h3>** (Hasta seis niveles de catalogación de títulos); **<header>** (Cabecera de la página, contiene los elementos introductorios); etc.

[3] "El Consorcio WWW, en inglés: *World Wide Web Consortium* (W3C), es un consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento de la *World Wide Web* a largo plazo." [7]

3. *Agrupación de contenidos:*
   Estructuran los párrafos y la muestra de contenido.

   <p> (Nuevo párrafo); **<ol>** (Lista ordenada); **<ul>** (Lista desordenada); ** ** (división genérica de contenido); **<table>, <tr>, <td>** (Tabla de contenidos); etc.

4. *Semántica a nivel de texto:*
   Añaden atributos, referencias y modificaciones visuales a elementos textuales.

   <a> (Insertar hiperenlace); **** (destacar importancia); **<time>** (Insertar valor de fecha y hora); **** (Inserción de atributos genéricos); etc.

5. *Contenido incrustado:*
   Inserta contenido no exclusivamente textual: imágenes, objetos genéricos, audios, vídeos, etc.

   <img> (Imagen); **<math>** (Fórmula matemática); **<source>** (Recursos multimedia); etc.

6. *Formularios:*
   Define y cataloga contenido introducido por el usuario: respuestas, imágenes, búsquedas, textos y posts…

   <form> (Representa un formulario, que es el conjunto de información que puede enviar el usuario a la página); **<label>** (Un control individual del formulario); **<button>** (Botón); **<textarea>** (Introducción de texto); etc.

Un posible fragmento de código que reúne un conjunto básico de las etiquetas más ejemplificantes es el siguiente:

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Ejemplo HTML</title>
	</head>
	<body>
		<h2>Título medio</h2>
		<p>
			<span id="intro"> Párrafo <strong> introductorio
			</strong>  </span>
		</p>
		<div class="lista">
			<ul>
				<li> Primer ítem de una lista no ordenada
				<li> Segundo item
			</ul>
		</div>
		<script src="utils.js"></script> <!-- Esto es un comentario HTML.No
		se ejecuta, y sirve para indicar al lector del código que aquí se
		está ejecutando código JavaScript del fichero utils.js -->
		<p> <img src="./image.png" weight="120" height="200"> ADIÓS </p>
	</body>
</html>
```

Figura 4. Ejemplo de uso código HTML

Leído por el navegador, el código muestra lo siguiente:

Figura 5. Ejemplo de uso HTML.

HTML respecto a Emendábile:

En el proyecto, se usará HTML para implementar la interfaz con el usuario. Es decir, se le asignarán las siguientes funciones:

- *Estructuración.* La página ha de tener una forma coherente, intuitiva y lógica.

- *Indexación de contenido visual*. Será necesario que la página muestre el vídeo capturado y el examen corregido, luego deberá poder incluir elementos multimedia compatibles con las librerías a utilizar. Esto se hará mediante *Canvas*, elemento que permite la creación y muestra de recursos visuales.

- *Interacción con el usuario.* El código HTML debe dotar a la página de los elementos de interacción necesarios para el correcto funcionamiento: introducción de información del examen (formularios), botones, muestra de exámenes, lectura de la web-cam, correcciones, etc.
  Además, se deberá comunicar con el código JavaScript para que reaccione correctamente a toda la información recibida.

- *Perfeccionamiento visual.* Ampliando su capacidad con CSS, el código HTML debe proveer a la página de un atractivo visual sugerente y diferenciable.

3.1.2.2 CSS

CSS, con traducción literal “Hojas de Estilo en Cascada” [10],es un lenguaje de programación cuya función es definir propiedades representativas a elementos externos, principalmente a elementos de documentos HTML.

Se programa en un documento .css externo, con referencias a los elementos HTML internos, y se importa el documento como hoja de estilos.

Por ejemplo, tomando de base el código HTML tomado anteriormente, se añade la siguiente hoja de estilos (**Figura 6**), en la que se asignan cualidades al fondo, texto, la imagen y la lista. El resultado se observa en la **Figura 7.**

```css
body{
        background-color: rgb(195, 255, 245);
}

.lista{
        background-color: rgba(0,0,0, 0.8);
        color: white;
        font-weight: bold;
        border: 3px solid;
        max-width: 25%;
        max-height: 70%;
        overflow: auto;
}

img{
        border-style: solid;
        border-color: rgb(254, 239, 86);
        margin-left:7%;
        font-color: rgb(195, 255, 245);
}

#intro{
        font-family: arial;
        font-style: italic;
        margin-left:2%;
}
```

Figura 6. Código de ejemplo lenguaje CSS.

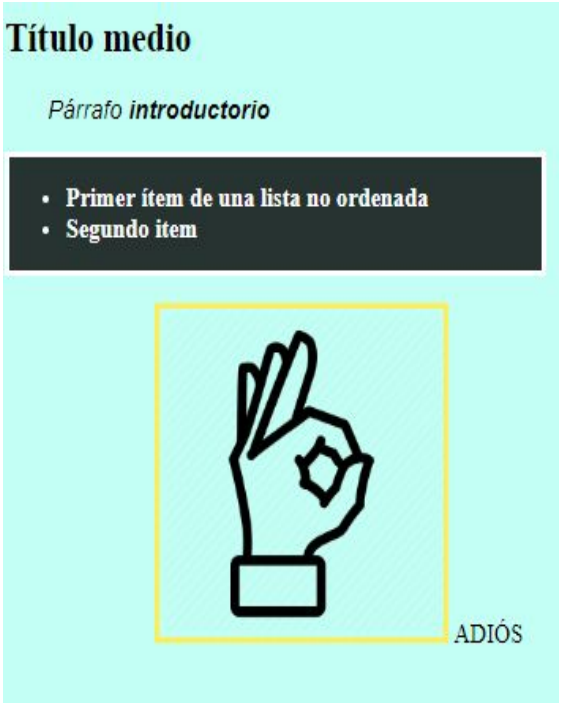

Figura 7. Perfeccionamiento visual con CSS.

3.1.2.3 JavaScript

JavaScript [11] es el lenguaje principal en programación Web del lado del cliente. Su uso es amplio y variado. Desde verificación y procesamiento de respuestas hasta creación y diseño de interfaces.
Su principal cualidad, que ofrece una expansión de uso sobresaliente, es la modificación dinámica del contenido web, de sus acciones y estilo.

Cómo se ejecuta JavaScript:

La ejecución del código es secuencial. Cuando se carga la página web, el entorno asimila el código fuente y lo representa visualmente. “Es como una fábrica que coge la materia prima (Las líneas de código) y lo presenta como el producto final (la página Web).” [12]

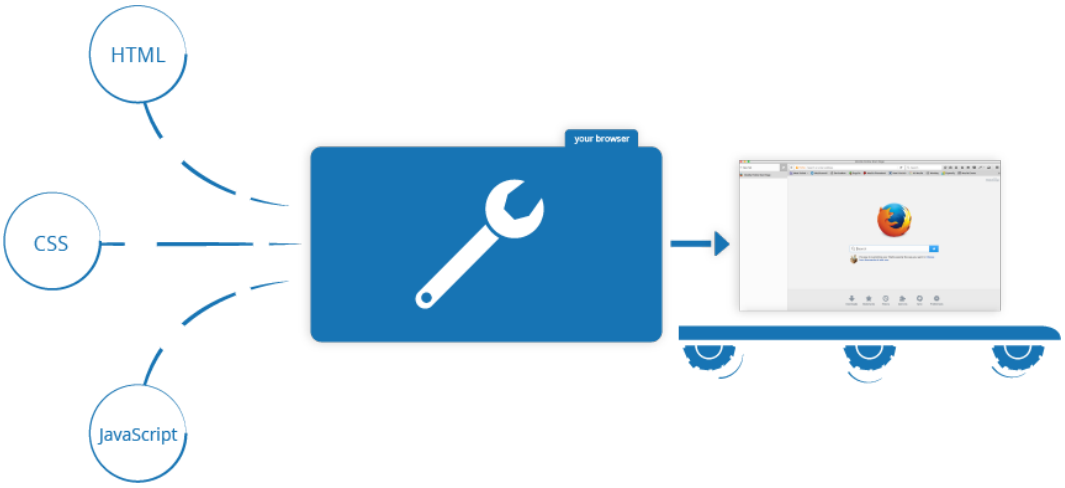

Figura 8. Ejecución secuencial del código web. [12]

Es decir, hasta que el código HTML y CSS no se ha congregado no se ejecuta el de JS, que servirá para la modificación y ampliación dinámica del código ya procesado.

Respecto al proyecto:

En Emendábile, JavaScript será en lenguaje principal, en el que se diseñarán e implementarán todas las funciones de captura, detección, procesado, comparación y corrección.
Como se explicará en la sección de Diseño, el lenguaje deberá ser capaz de la asimilación y manipulación de imágenes, trabajo con arrays, instanciamiento de objetos y funciones, interacción directa con el usuario y modificación inmediata de contenido estático.
Para ello se hará uso de librerías externas de visión artificial y muestra de contenido multimedia.

3.2 Visión Artificial

3.2.1 Introducción a la Visión Artificial

El segundo pilar sobre el cual se edifica el desarrollo del proyecto es la visión artificial, definida como la transformación de una imagen o vídeo recibido en una decisión o nueva representación [13].

Por ejemplo, el sistema puede tomar la *decisión* de que en la imagen analizada hay un coche, o una persona. O puede generar una *nueva representación*, semejante a la imagen original pero en blanco y negro. Ambas funcionalidades se atribuyen al concepto de Visión Artificial.

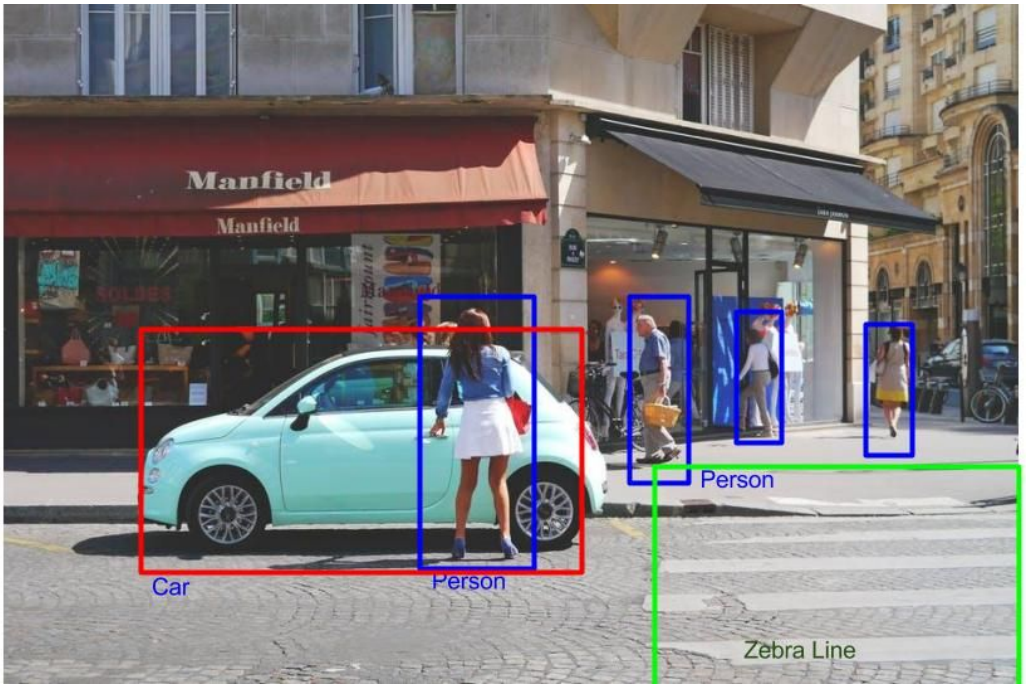

Figura 9. Ejemplo de decisión en Visión Artificial. [14]

Figura 10. Ejemplo de nueva representación en Visión Artificial. [15]

En el proyecto, si se toman como únicas herramientas los lenguajes de programación detallados con anterioridad (HTML, CSS, JavaScript), no se tiene la posibilidad de implementar un sistema de visión artificial. Ya que operar dichas transformaciones en las imágenes requiere de un coste de elaboración demasiado elevado.

Por ello, será necesario el uso de librerías externas [4] que amplíen las capacidades visuales del sistema.

A continuación, se expondrán los requisitos que debe cumplir dicho sistema externo y compararemos y decidiremos entre las diferentes alternativas.

3.2.2 Requisitos del sistema de Visión Artificial

Para que se pueda implementar correctamente el programa, el sistema de visión artificial que se utilice debe incluir un conjunto de funciones básicas e imprescindibles.
Analizando el código de Eyegrade, y diseñando la aplicación conforme se explica en la sección de *Diseño*, se halla que las funciones imprescindibles son las siguientes:

1. *Cambio a escala de grises.*
   Para la mejor detección de líneas, y por lo tanto mayor eficiencia en tiempo de cómputo, será necesario preprocesar la imagen inicial. Una componente de este preprocesado es transformar la imagen a escala de grises. **Figura 10.**

2. *Transformación a binario.*
   Se trata de otra operación del preprocesado. Transforma todos los valores de la matriz imagen a binario (pixel negro o blanco), así el contraste aumenta y la detección es más sencilla. **Figura 11.**

3. *Detección de líneas.*
   Una de las funciones principales. La librería ha de ser capaz de reconocer patrones representativos de rectas. **Figura 12.** Esta función será la base de la detección, ya que determina la localización y naturaleza de las celdas y tablas.

4. *Dibujo de rectas en imagen.*
   Será necesaria para el procesamiento de la imagen. Una vez detectadas las líneas se deberán dibujar para comparar, fusionar, determinar intersecciones, etc.
   Figura 12. (Una vez detectadas, las líneas se dibujan en una imagen vacía)

5. *Dibujo de círculos en imagen.*
   En la muestra del examen corregido se indica la casilla marcada con un círculo de color dependiente de la calidad de la respuesta. Esta función es más prescindible pero otorga de vistosidad y atractivo al programa. **Figura 13.**

[4] "En informática, una biblioteca o **librería** es un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca." [16]

6. *SVM*
   A disposición del proyecto, por elaboración del equipo de *Eyegrade*, en el desarrollo se contará con una máquina SVM entrenada para el reconocimiento de dígitos.
   Luego, para la detección automática del identificador del alumno será necesario importar los datos de dicha SVM a través de la librería externa.

7. *Operaciones con imágenes en forma matricial.*
   El objeto sobre el que se guarde la imagen capturada debe ser una matriz de pixeles, y ésta debe contar con métodos propios de cálculo.
   Esto se utilizará, por ejemplo, para la comparación de imágenes en la detección de las marcas.

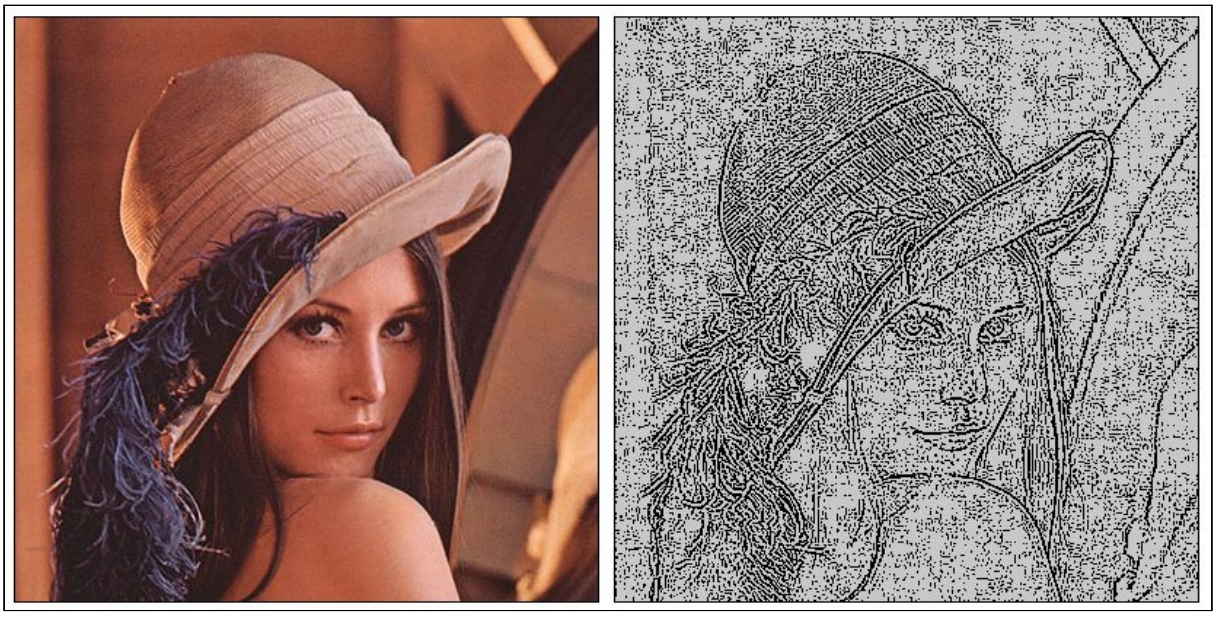

Figura 11. Transformación a binario (Adaptive Thresholding). [18]

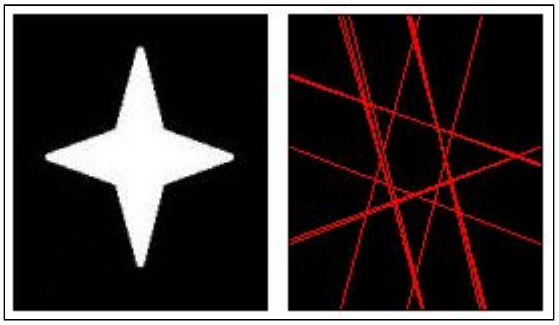

Figura 12. Detección de rectas en imagen. [18]

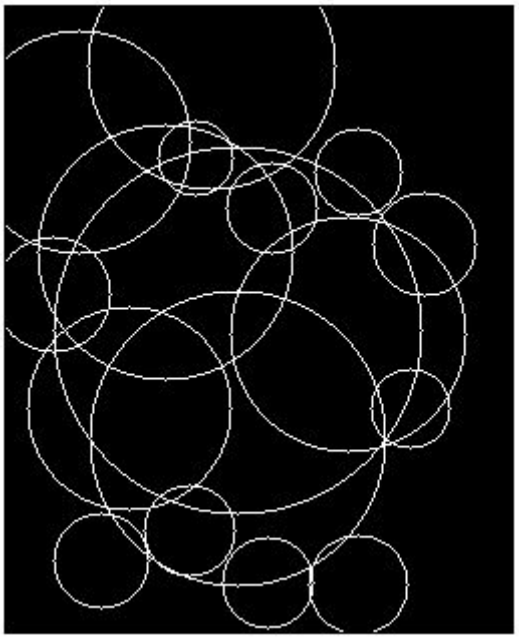

Figura 13. Dibujo de círculos sobre imagen vacía. [18]

Además, como requisitos externos a la funcionalidad directa del programa, están:

a) *Compatibilidad con JavaScript*

   La librería externa debe ser implementable y accesible desde JavaScript en un entorno web. Esto limita bastante las posibilidades, ya que la mayoría de librerías de visión artificial están pensadas para lenguajes de análisis de datos masivos. (Python, Ruby, Matlab, ...)

b) *Compatible con Eyegrade*

   Ya que el proyecto está basado en Eyegrade, y que muchas de las funcionalidades necesarias son comunes, para una mayor compatibilidad sería óptimo un sistema de visión artificial semejante al usado en el proyecto de referencia.

c) *Licencia de código libre*

   Debido al contexto de creación del proyecto y a la baja disponibilidad frente a recursos comerciales o privativos, la naturaleza de la librería a utilizar debe ser de *open-source.*

3.2.3 Decisión

Teniendo en cuenta los requisitos vistos, tanto de funcionamiento como de compatibilidad y disponibilidad, se obtiene que la mejor opción a escoger es la librería OpenCV, por los siguientes motivos:

- **Compatibilidad con Eyegrade.**

  Eyegrade, programado en Python, hace uso de este misma librería, OpenCV. Este es un factor muy importante en la elección, puesto que las funciones a utilizar y los procedimientos básicos serán lo más semejantes posible. Sin embargo, existirán, por supuesto, cualitativas y notables diferencias entre los lenguajes de programación.

- **OpenCV y JavaScript.**

  OpenCV, con primera versión alfa en 1999, ha ido actualizando y amplificando su alcance y contenido. Es desde la versión 3.0 desde la que han ampliado su compatibilidad y añadido como lenguaje disponible a JavaScript, a través de OpenCv.js. Esto supone la ventaja principal frente a otros sistemas, sin la cual la librería sería completamente inútil.

- **Licencia BSD.** [17]

  OpenCV cuenta con licencia BSD, es decir, licencia que suministra al proyecto todos los servicios necesarios gratuitamente.

3.2.4 OpenCV

Como ya se ha introducido en la sección anterior, OpenCV [13] es es una librería de visión artificial de código abierto. Su origen se remonta a 1999, año en el que se lanza su versión alfa de mano de Intel[5]. Desde entonces se ha sometido a un continuo de actualizaciones que han derivado en:

- Más de 500 funciones entorno a la visión artificial: seguridad, interfaz, ajuste, procesado, etc.
- Posible implementación desde multitud de lenguajes. A pesar de estar programado en C/C++, es accesible para plataformas en Python, Ruby, Matlab, Java, Javascript…
- Desarrollo máximo y global. OpenCV ha sido utilizado por multitud de empresas para objetivos como: construcción de imágenes de satélite, decisión en programas visuales médicos, seguridad, vehículos subacuáticos, etc.

3.2.4.1 OpenCV.js

En momentos en los que la web se ha convertido en la plataforma de desarrollo más versátil y común, la necesidad de una librería que introduzca servicios de visión artificial en este entorno se amplifica.
Es por esto por lo que Intel Corporation y posteriormente el programa de Google Summer de 2017 financiaron la ampliación de la librería OpenCV a plataformas web vía Javascript.

Para ello se hace uso de Emscripten, un compilador a JavaScript. Partícipes son asm.js o WebAssembly, formatos de compilación de alta velocidad para el navegador web.
De esta modo se consigue suministrar al navegador de un conjunto de funciones básicas de OpenCV que de momento puedan suplir las necesidades que surjan ante el desarrollo web.

Sin embargo, tiene un inconveniente: debido a su reciente e inacabada implementación, OpenCV.js no cuenta aún con todas las funcionalidades necesarias para el comportamiento óptimo del programa. La ausencia más significativa es no contar, al contrario que la versión principal de OpenCV, con una implementación de máquinas SVM, lo que impide a priori la detección y lectura del identificador del alumno.

[5] Intel Corporation es el mayor fabricante de circuitos integrados del mundo. Fue fundada en 1968 como **Int**egrated **El**ectronics Corporation

3.3 Eyegrade

Eyegrade [1] es un sistema de corrección automática de exámenes de opción múltiple. Su objetivo principal es ser un recurso abierto, de bajo coste material (sólo una webcam), que facilite la corrección de exámenes tipo test a cualquier docente que procure sus servicios.

Eyegrade se distribuye con licencia *open-source*, luego está permitida la referenciación y uso de contenido para elaboración de nuevos proyectos fuera de la marca.

En este análisis se definirá su funcionamiento, estructura, y el conjunto de funciones capitales que han sido fundamentales para el desarrollo de Emendábile.

3.3.1 Eyegrade: Funcionamiento

Eyegrade tiene tres funcionalidades principales:

1. Corrección automática de exámenes. Detecta cada prueba y la evalúa, atribuyendo a cada alumno su nota correspondiente.

2. Exportación de notas. Permite exportar en formato CSV el conjunto de notas resultante del proceso de corrección.

3. Creación de modelos. Concede la posibilidad al docente de crear un modelo de examen según el cual se pueda llevar a cabo la corrección.

Con estas tres funcionalidades se alcanza un programa de corrección capaz de cumplir con su objetivo de manera completa.

De ahora en adelante, por eficiencia en el vocabulario empleado, el texto se referirá a las siguientes connotaciones de cada palabra:

- *Modelo o sesión:* examen diseñado por el profesor, con su conjunto de preguntas y respuestas válidas.
- *Tipo:* cada modelo cuenta con varios tipos. Las soluciones son las mismas, pero sus distribuciones no, es decir, varía el orden de las preguntas y de las soluciones dentro de cada una de éstas.
- *Examen / prueba:* con esto será referido cada uno de lo tests rellenados por cada alumno. Cada uno de éstos tiene asociado un modelo, un tipo, una nota y un alumno.

En el uso, el funcionamiento de Eyegrade es el siguiente:

Eyegrade basará su corrección en un formato determinado de cuadrículas de solución, tal y como se muestra en la **Figura 14**. Si el examen no se basa en este formato no es compatible con la corrección de Eyegrade.

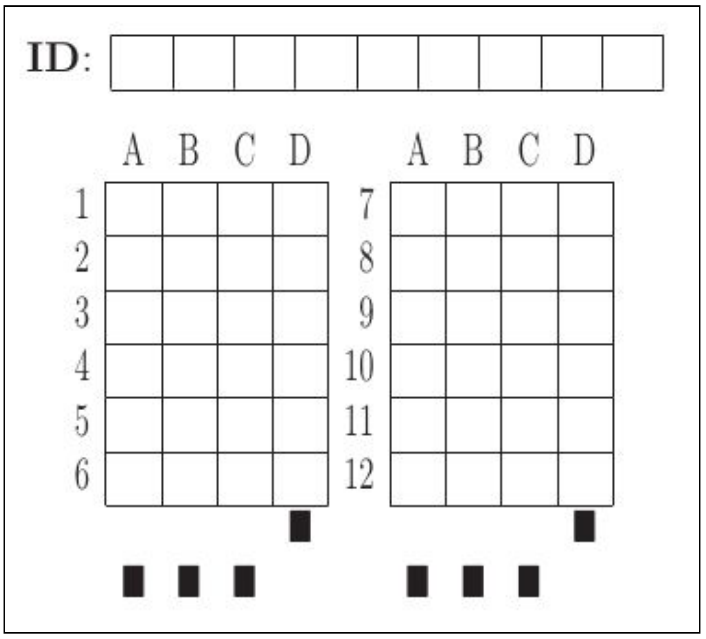

Figura 14. Formato de examen para la corrección en Eyegrade

Eyegrade es un programa de corrección orientado a modelo. Es decir, cada modelo se evalúa de forma aislada al resto. Luego, lo primero que se solicitará para la corrección será la introducción de un modelo, ya sea manualmente o por importación. (**Figura 15**)

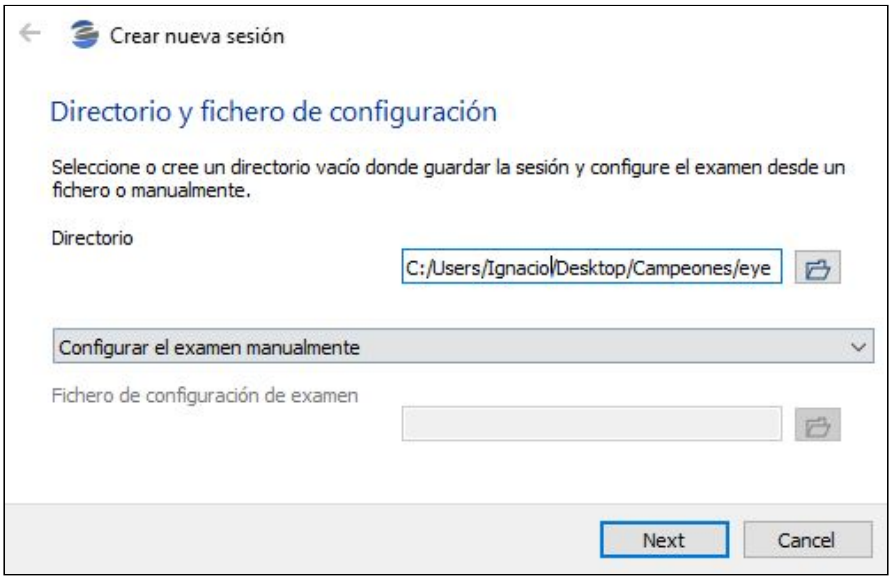

Figura 15. Agregación de modelo o sesión en Eyegrade

La agregación manual del modelo implica la introducción de parámetros como número de preguntas, opciones por pregunta, puntaje, soluciones, tipos, etc.

Se debe añadir también el conjunto de alumnos con sus identificadores. De esta manera el programa podrá asignar las notas correspondientes a cada alumno acorde a la detección de cada prueba. (**Figura 16**)

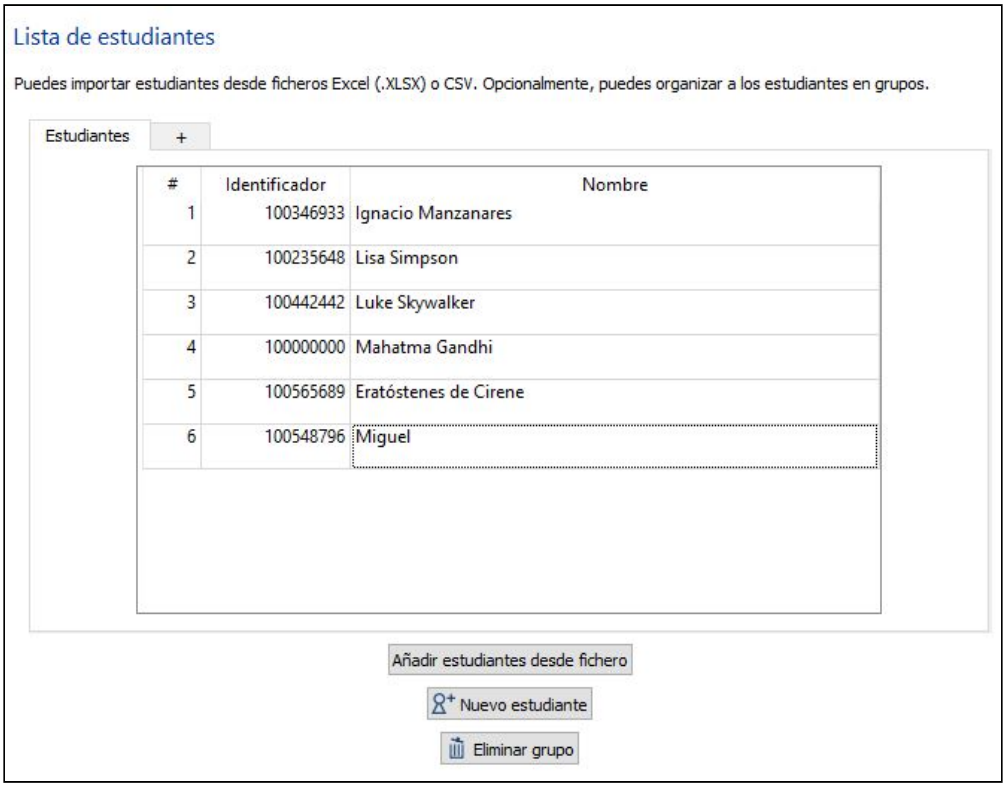

Figura 16. Agregación de estudiantes para la posterior asignación de nota.

Como se observa en la **Figura 16**, se pueden agregar manualmente o importar como subconjunto desde un fichero externo.
Una vez introducidos todos los parámetros necesarios para la corrección y asignación de nota, puede comenzar la corrección de cada examen, que sigue el siguiente procedimiento:

1. Comienzo de grabación y colocación del examen. Se le indica al programa que comience la corrección. Si la casilla de respuestas es completamente visible por la webcam, el programa asignará directamente la nota al alumno correspondiente en la base de datos.
2. Revisión de corrección. Una vez detectado y corregido el examen, el programa muestra por pantalla el resultado obtenido, señalando las casillas marcadas como correctas o incorrectas. El usuario puede corregir el resultado del programa en caso de equivocación, indicando las casillas marcadas.
3. Cambio de examen. Si el resultado del programa no requiere de más corrección, se guarda la nota del alumno y se continúa con el siguiente examen con el botón “Continuar”.
4. Sucesión indefinida de los pasos 2 y 3 hasta finalizar el conjunto de exámenes.
5. Finalización. Se le indica al programa que no quedan más exámenes por corregir, y, si se quiere, se exporta el conjunto de notas resultante.

Una vez realizados estos pasos, el docente tendría las notas de sus alumnos en un formato cómodo y versátil.

3.3.2 Eyegrade: Diseño.

Eyegrade tiene una estructura modular. Cada uno de los módulos principales desempeña unas funciones concretas y parcialmente independientes.
Sin embargo, Emendábile solo toma como referencia en su programa el módulo más importante de Eyegrade: la detección.

Eyegrade: Detección

Este módulo del programa es el contiene las funciones más importantes y diferenciables de Eyegrade. Su objetivo es la captación de las casillas marcadas, el identificador del alumno y el tipo de examen.

El procedimiento por el cual funciona sigue los siguientes pasos:

1. *Preprocesado de la imagen:*

   Para poder llevar a cabo una mejor detección de las líneas, cruces y dígitos, se ha de filtrar la imagen en su color. Para ello, usando la librería OpenCV se aplicarán las siguientes modificaciones a la imagen inicial:

   a) Transformación a escala de grises.

   b) Transformación a binario. Mediante otra sencilla función de OpenCV (binary thresholding) se binariza el color de la función a partir de un umbral en cada pixel. De este modo obtenemos contraste máximo entre área marcado y vacío.

2. *Detección de rectas:*

   Se llama al método *HoughLines* de OpenCV, que en función de un determinado parámetro de sensibilidad obtiene un conjunto de rectas detectadas en la imagen.

   Dependiendo del valor de dicho parámetro, puede ocurrir que una línea no se detecte, se detecte normalmente o se detecte como múltiples líneas cercanas entre sí. Si ocurre este último caso, un filtrado posterior puede combinar todas ellas en una única línea. Este parámetro de sensibilidad se irá modificando según el éxito de la detección en la imagen.

   Las rectas resultantes se obtienen en el formato [θ, ρ] donde ρ es la distancia perpendicular de la recta al origen y θ es el ángulo de esa perpendicular respecto al eje horizontal. (**Figura 17**)

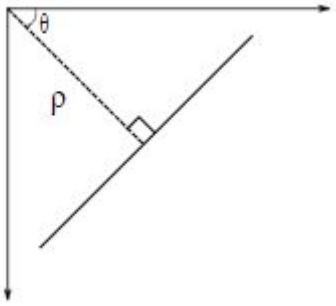

Figura 17. Formato rectas HoughLine [19]

3. *Detección de direcciones y ejes:*

   Una vez se ha obtenido un conjunto de rectas detectadas en la imagen, se han de determinar las direcciones principales y ejes, que actuarán como base de todas las operaciones geométricas.

   Estos ejes tendrán el siguiente formato: (θ_E, [(θ_L, ρ)]). Cada eje contiene información de su inclinación fundamental (θ_E) y del subconjunto de líneas detectadas asociadas por proximidad a esa dirección ([(θ_L, ρ)])

   Como la imagen enfocada debe ser una cuadrícula de corrección, el objetivo será obtener solo dos ejes, el horizontal y el vertical.
   Para la obtención de estos ejes se realizará un filtro a todas las rectas, que consistirá en comparar con una inclinación de referencia. Si está lo suficientemente próxima se añadirá dicha recta al eje de inclinación de referencia.

   Si se obtienen más de dos ejes se eliminarán los cruzados comparando las inclinaciones (π/2 y π serán los referentes de esta comparación).
   Si se obtienen menos de dos ejes se rechaza la imagen y se repite la captura con otro parámetro de sensibilidad.

4. *Filtrado de rectas:*

   Una vez se tienen los dos ejes, horizontal y vertical, se filtrarán las líneas para simplificar a una línea objeto por cada línea real de la cuadrícula. Para ello se recorrerá todo el conjunto de líneas de un eje.
   Si una línea es más cercana a la anterior que cierto parámetro umbral, se junta con las anteriores realizando la media de su distancia e inclinación. Si no lo es, se establece como nueva línea de comparación frente a las siguientes. Ese parámetro determina la rigidez de filtrado y se determina por comprobación de funcionamiento.
   Como resultado de este paso se deben obtener dos ejes, horizontal y vertical, cada uno de ellos con el número de líneas de esa inclinación reales en la cuadrícula.

5. *Detección de celdas:*

   Teniendo el conjunto de rectas resultantes del paso anterior se buscará hallar las coordenadas de las celdas. Para eso se aplicarán intersecciones entre todas las rectas. El resultado es una matriz de esquinas. Cada una de ellas definida con un duplo en formato (CoorX, CoorY).

   Posteriormente se procesa esta matriz de esquinas para dar lugar a una matriz de celdas. Para ello se define cada celda como un objeto con cuatro atributos: sus esquinas. Y éstas a su vez como duplos de coordenadas.
   De esta manera, el formato de cada celda es el siguiente:

   celda.plu[6] = $(x_{left-up}, y_{left-up})$ celda.pru = $(x_{right-up}, y_{right-up})$

   celda.pld = $(x_{left-down}, y_{left-down})$ celda.prd = $(x_{right-down}, y_{right-down})$

6. *Detección de respuestas:*

   Una vez halladas las celdas se deberá detectar la respuesta del alumno. Es decir, la X que indica la decisión del examinado. Para ello se hace uso de una máquina entrenada SVM.
   Este modelo está entrenado para el reconocimiento de cruces en casillas. Su implementación y entrenamiento es bastante sencillo.
   Una vez pasadas todas las celdas por el modelo, se juntan todas las respuestas en la matriz de decisiones. Matriz que se usará para la corrección del examen.

7. *Detección de tipo de examen:*

   Para poder llevar a cabo la corrección es necesario conocer el tipo de examen, ya que las soluciones correctas varían de posición. Por eso el formato de corrección debe añadir unas cuadrículas inferiores, denominadas *infobits,* que revelan el tipo de solución:

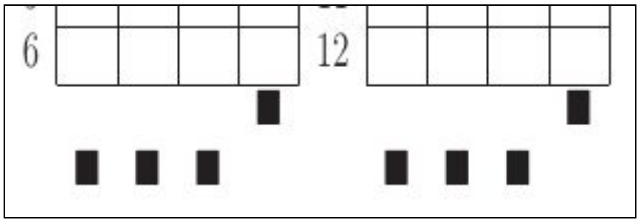

Figura 18. Infobits. Detección del tipo de examen

[6] La nomenclatura *plu* significa “point left up”, indicando posición horizontal y vertical. Del mismo modo ocurre con el resto de componentes.

Los infobits indican el tipo de examen, para ello siguen el procedimiento:

- Si se coloca arriba su valor es 1, si se coloca abajo su valor es 0.
- Los tres primeros bits indican en binario el tipo de examen (Máximo 8 tipos), el cuarto bit se determina por una función de redundancia de los 3 anteriores, y a partir del cuarto se repite la secuencia.

Por lo tanto, si en la detección no se cumplen las propiedades de redundancia o repetición se descarta la imagen.

Para detectar los infobits se analiza el área inferior a cada última casilla. Se compara la cantidad de píxeles oscuros de dicho área con una matriz de semejante tamaño, pero vacía. Si la diferencia supera cierto parámetro de sensibilidad entonces se supone que el área contiene un infobit activado.

8. *Detección de identificación del alumno:*

   Para la detección del número de identificación del examinado se llevará a cabo el mismo proceso que para la detección de cruces en cada casilla. Es decir, se utilizará un modelo entrenado mediante SVM de reconocimiento de dígitos.
   Para ello en la creación del modelo de examen se ha de haber introducido el número de dígitos del identificador.
   Una vez sabido el número de dígitos y la posición de los mismos sólo se debe llamar al modelo entrenado para que decida cada uno de los números.

Una vez asimilados todos los datos detectados por este módulo, Eyegrade hace uso de ellos llevando a cabo la corrección usando otros módulos y subsistemas del programa.

Sin embargo, no analizaremos el resto de funcionalidades de Eyegrade, ya que no son referencia para nuestro proyecto. Y, debido al cambio de entorno (lenguaje, objetivo, ...), distan mucho de las funcionalidades que se implementarán en Emendábile.

3.4 Soluciones similares ya existentes

Como se ha expuesto en la introducción, el desarrollo tecnológico actual que busca informatizar tareas automáticas supone uno de los mercados que más movimiento tienen. Es por eso por lo que este proyecto no es único, y el ambiente en el que se desarrolla es propicio y fructífero.

Por eso existen muchas soluciones para el problema a resolver en este proyecto. Muchas empresas y desarrolladores han apostado por la corrección automática de exámenes multi opción o “tipo test”.

Sin embargo, Emendábile tiene cualidades que no se han encontrado en el mercado actual. Y que aportan al proyecto diferenciabilidad y potencial.
Estas cualidades son las siguientes:

1. *Sin necesidad de escáner*.

   El resto de programas encontrados hacen uso en lo total investigado de máquinas de escáner. [20] [21] [22] [23] [24]

   Esto supone un incremento de velocidad proporcional a la calidad y coste del escáner.
   Sin embargo, nuestro proyecto, y su referencia Eyegrade, tienen como objetivo la utilización *Low-Cost* o de bajo coste. Ya que para la utilización del producto sólo es necesario el uso de una webcam.

2. *Instantaneidad online.*

   Emendábile es un programa extraordinariamente inmediato. Sólo es necesario el acceso a la página web, introducir el modelo del exámen y su solución. Acto seguido comienza la corrección examen a examen.
   La mayoría de programas semejantes evaluados dependen del tiempo de escaneo, subida de archivos, disposición manual de alumnos y tratamiento posterior de datos.
   Su resultado puede ser más completo, pero no cuenta con el bajo coste y dinamismo de Emendábile.

3. *Código libre.*

   Al igual que Eyegrade, Emendábile se distribuye con licencias del código libre. Esto supone una mayor disponibilidad para todos los desarrolladores y usuarios que le den uso de alguna manera.

4. ARQUITECTURA DEL SISTEMA

Emendábile es un sistema web de ejecución en el lado del cliente. Su objetivo es la corrección automática de exámenes tipo test a partir de la webcam, y tiene como referente principal el método de detección de Eyegrade.

Para el desarrollo de Emendábile se habrá de realizar un diseño general de su arquitectura, definiendo sus módulos principales y las relaciones o nexos entre ellos.

La arquitectura básica del proyecto será la siguiente:

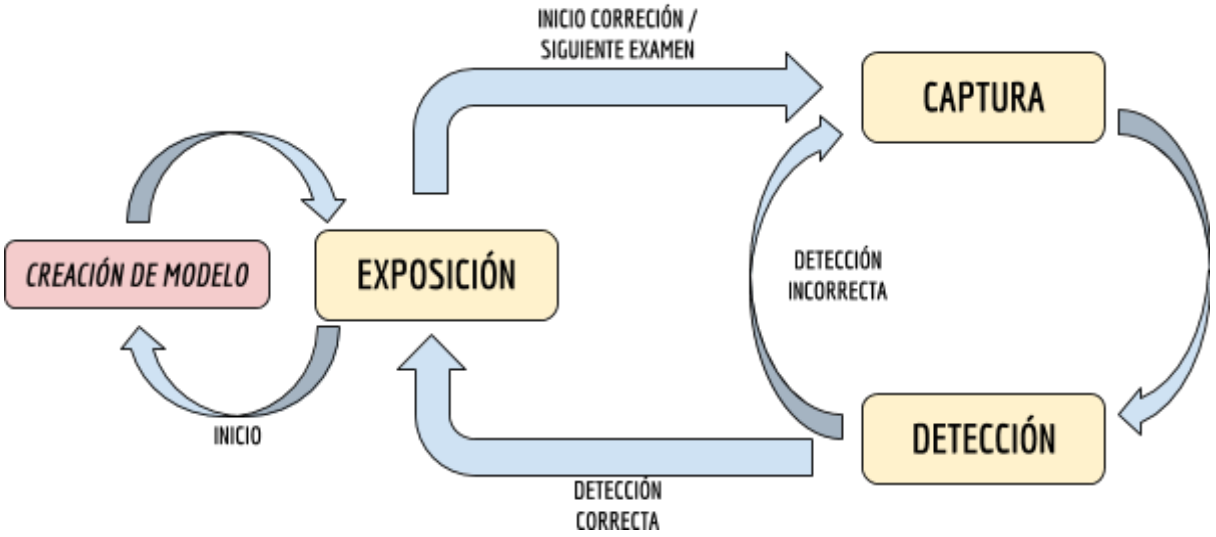

Figura 19. Arquitectura principal de Emendábile.

La arquitectura fundamental está compuesta por tres módulos principales: *exposición*, *captura* y *detección*, además de un cuarto submódulo, de *creación de modelo*, que forma parte del módulo de exposición.

A continuación, para entender mejor el desarrollo de la arquitectura de Emendábile, se expondrá su funcionamiento, aludiendo en cada función principal al módulo que gestiona y produce la acción.

5.1 Funcionamiento Emendábile

El programa se encuentra aún en una versión inicial, no preparada para su lanzamiento, por el gran desarrollo (sobre todo gráfico) que eso implicaría, y con numerosas posibles mejoras. Sin embargo, el funcionamiento implementado ya es suficiente para aprovechar su uso y que éste mejore cualitativamente la corrección de pruebas tipo test.

Dicho funcionamiento es el siguiente:

El usuario se conecta al servidor, entrando en la página web. La primera imagen que obtiene es la siguiente:

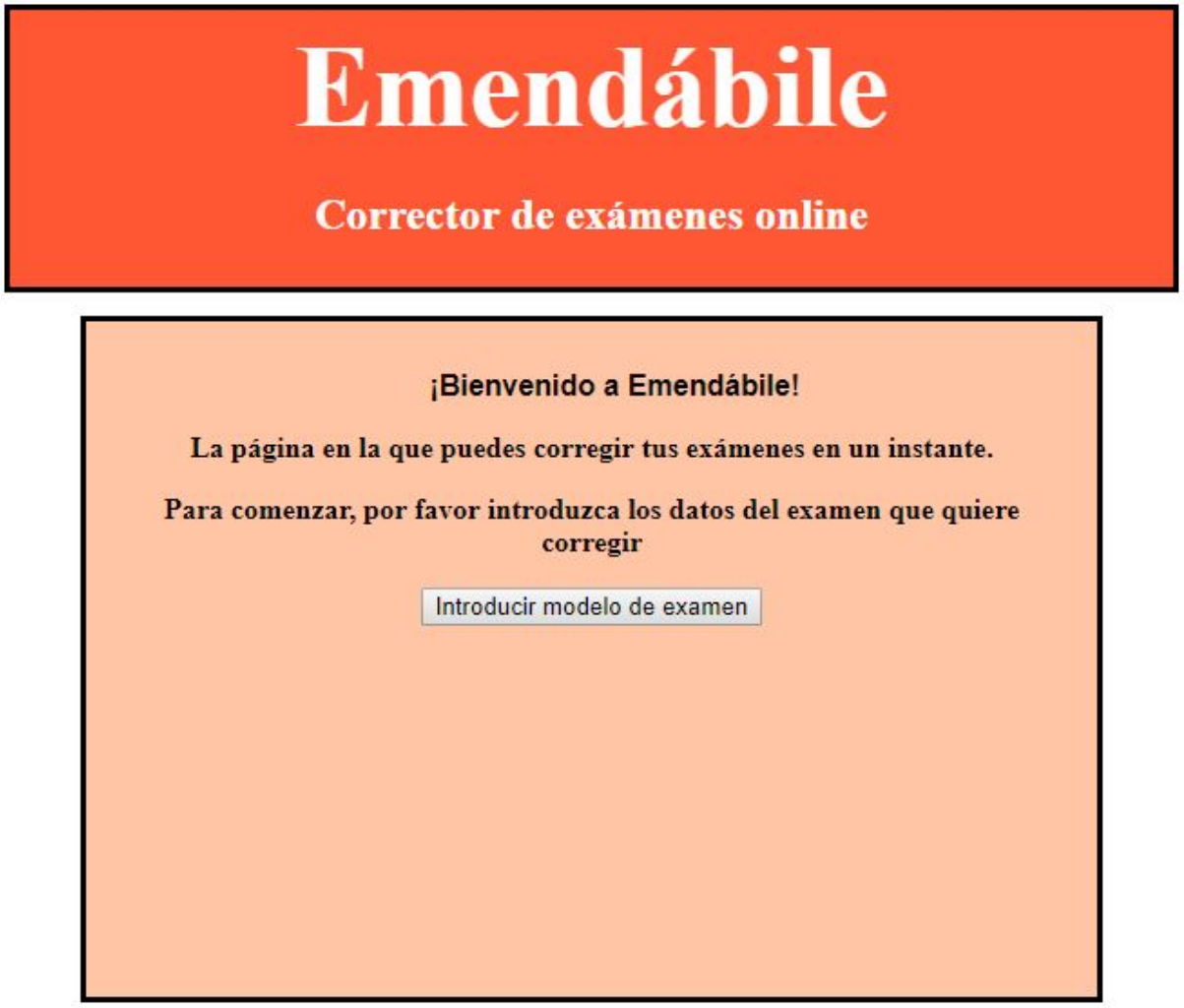

Figura 20. Interfaz de inicio Emendábile.

Esto formaría parte del módulo de Exposición, que gestiona la interfaz del usuario. Es decir, la exposición de las imágenes, las proposiciones de acción (botones, formularios), etc.

Como se ve en la **Figura 20**, una vez se ha entrado en la página, ésta ofrece la creación de un nuevo modelo de examen sobre el cual se puedan realizar las correcciones. De la creación del modelo se encarga por supuesto el módulo homónimo.

Para ello se rellenará el formulario de agregación de modelo (**Figura 21, Figura 22**) introduciendo el número de preguntas, número de opciones, puntaje positivo, puntaje negativo, número de tipos y soluciones de cada tipo.

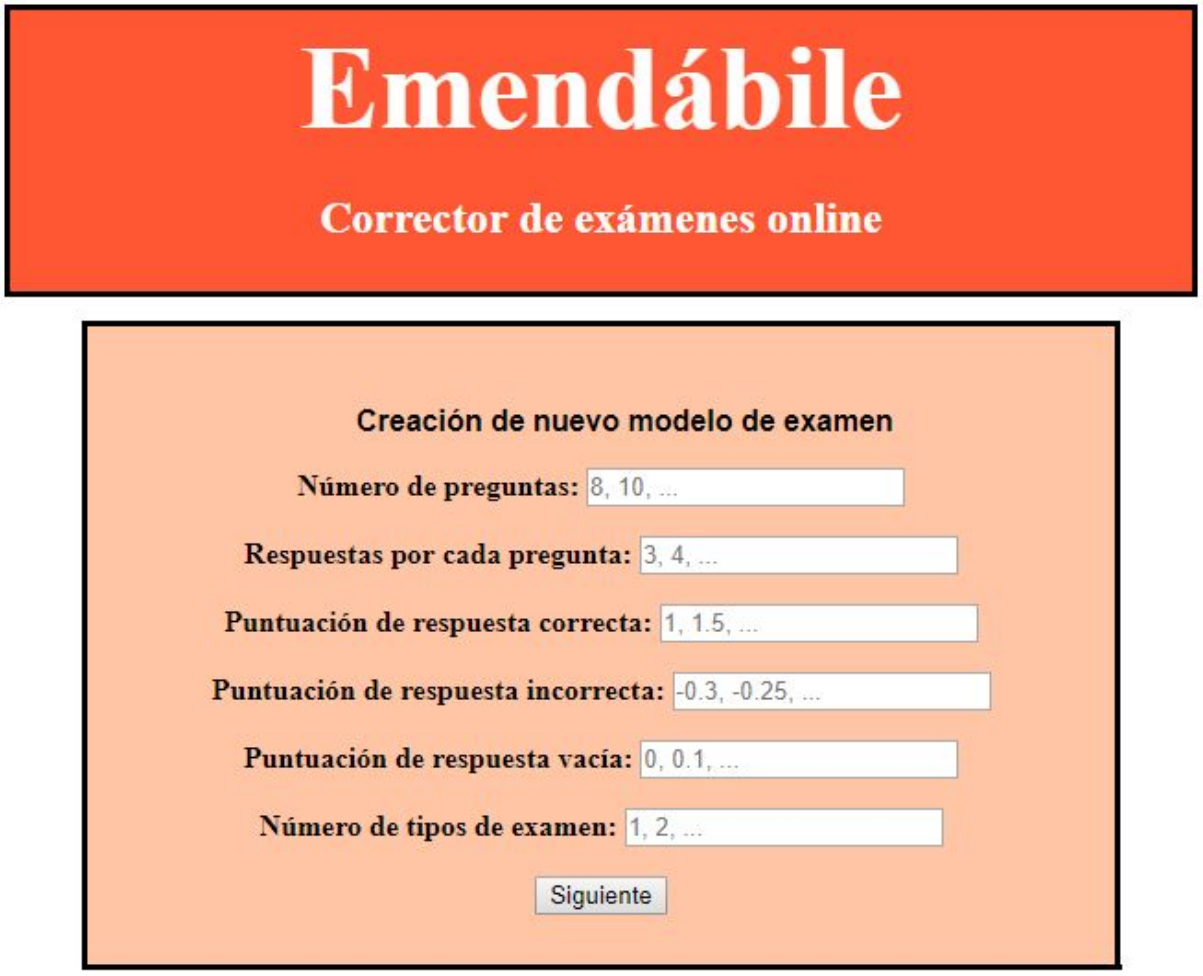

Figura 21.Introducción de nuevo modelo. Paso 1.

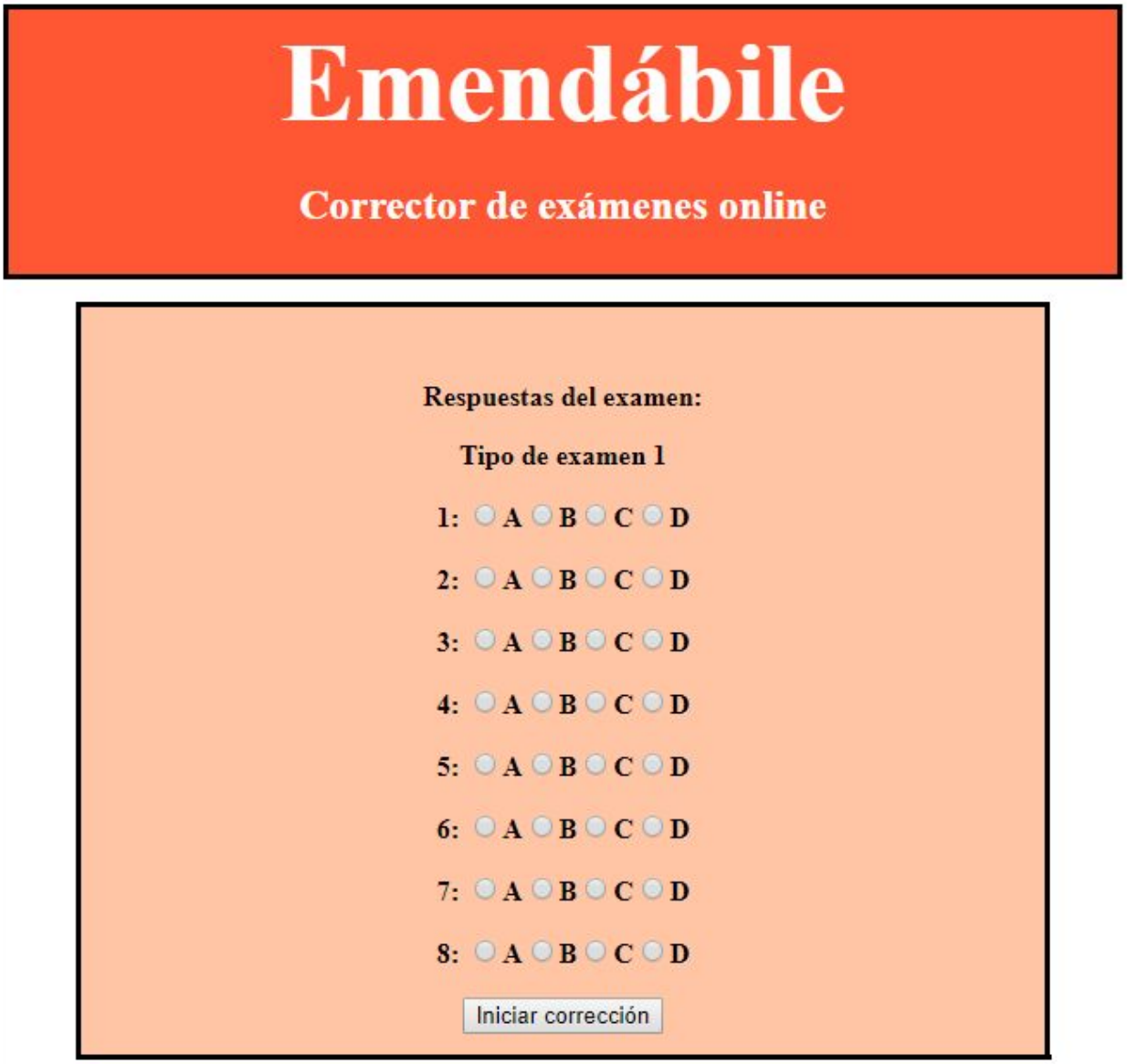

Figura 22.Introducción de nuevo modelo. Paso 2.

Una vez realizado cada paso del formulario de nuevo modelo, verificará las respuestas. Al finalizar, indicará que ya se puede empezar a corregir. Para ello, se dará al botón de “Iniciar Corrección”.

Esto comunica directamente con el módulo de captura. Éste accede a la cámara y transmite la imagen capturada al módulo de detección. Éste (el más importante del programa) se encargará de indicar las respuestas y tipo de cada prueba.
Si la imagen no está bien realizada, no corresponde a un examen en el formato adecuado, o se da algún error la captura se desecha y se vuelve a solicitar otra al módulo de captura.

Si la detección se efectúa correctamente, se comunican los resultados (decisiones, coordenadas de las celdas, tipo de examen) al módulo de exposición.

En exposición se mostrará el examen corregido, indicando con un círculo cada decisión del examinado, y con el color de ese círculo si la respuesta es correcta (verde) o incorrecta (rojo). Además se indicará la nota resultante del examen, teniendo en cuenta el peso de cada acierto, fallo y respuesta en blanco. (**Figura 23**).

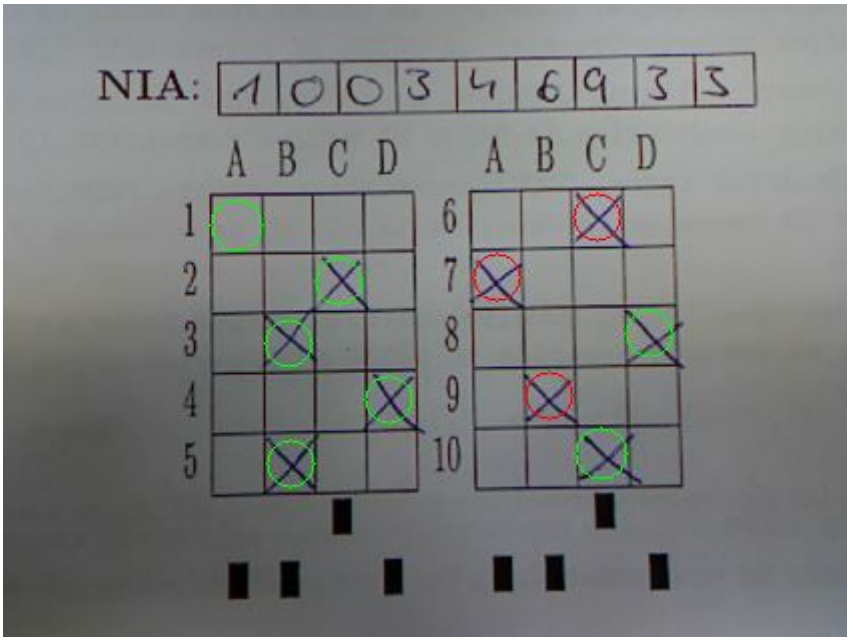

Figura 23.Muestra de examen corregido.

Una vez se ha mostrado el examen corregido, el propio módulo de exposición ofrece la posibilidad de revisar y rectificar la corrección realizada. Por ejemplo, en el caso mostrado en la **Figura 23**, el programa ha detectado una respuesta que no se ha realizado en la pregunta número 1.

Para solucionarlo, de forma intuitiva, se indica la casilla a corregir con el ratón, y el programa rectifica la detección y nota. (**Figura 24**)

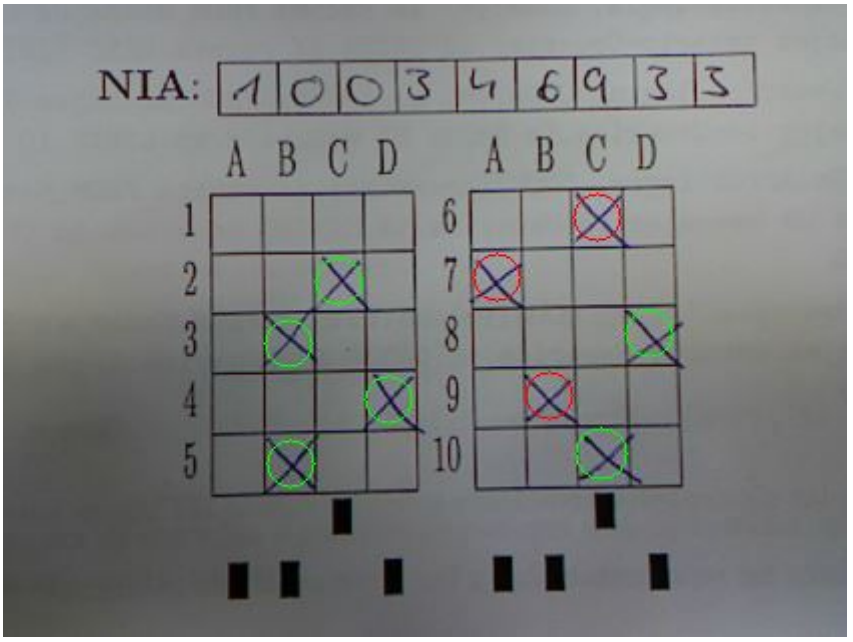

Figura 24. Rectificación de detección.

Cuando se considere que la corrección del examen es correcta, ésta se finaliza, y se procede de dos posibles maneras: “siguiente examen” o “finalizar corrección”.

Para la versión realizada del programa, al contrario que en Eyegrade y por razones que se expondrán más adelante, todavía no se puede guardar la nota para cada alumno en una lista exportable.

5. DISEÑO E IMPLEMENTACIÓN

Expuesto ya el funcionamiento y arquitectura principal del programa, se analizarán ahora con detalle cada uno de los módulos. Para ello, se realizará una panorámica de su funcionamiento, que posteriormente se detallará examinando sus métodos y procedimientos internos.

5.1 Exposición

El módulo de exposición es el encargado de la interacción con el usuario. En él se desarrollan la interfaz gráfica, la visualización de la webcam, y la representación, rectificación, inicio, cambio y finalización de la corrección. Incluye además dentro de su arquitectura el submódulo de creación de modelo, como se detallará más adelante.

Ahora, a partir de la exposición de la interfaz gráfica, analizaremos cada uno de los componentes que se han desarrollado.

Para el diseño y desarrollo de la interfaz gráfica se usarán las herramientas indicadas en el apartado 3.1.2: HTML, CSS y JavaScript. El objetivo será crear una visualización sencilla e intuitiva.

La siguiente estructura cumple dicho cometido: (**Figura 25**)

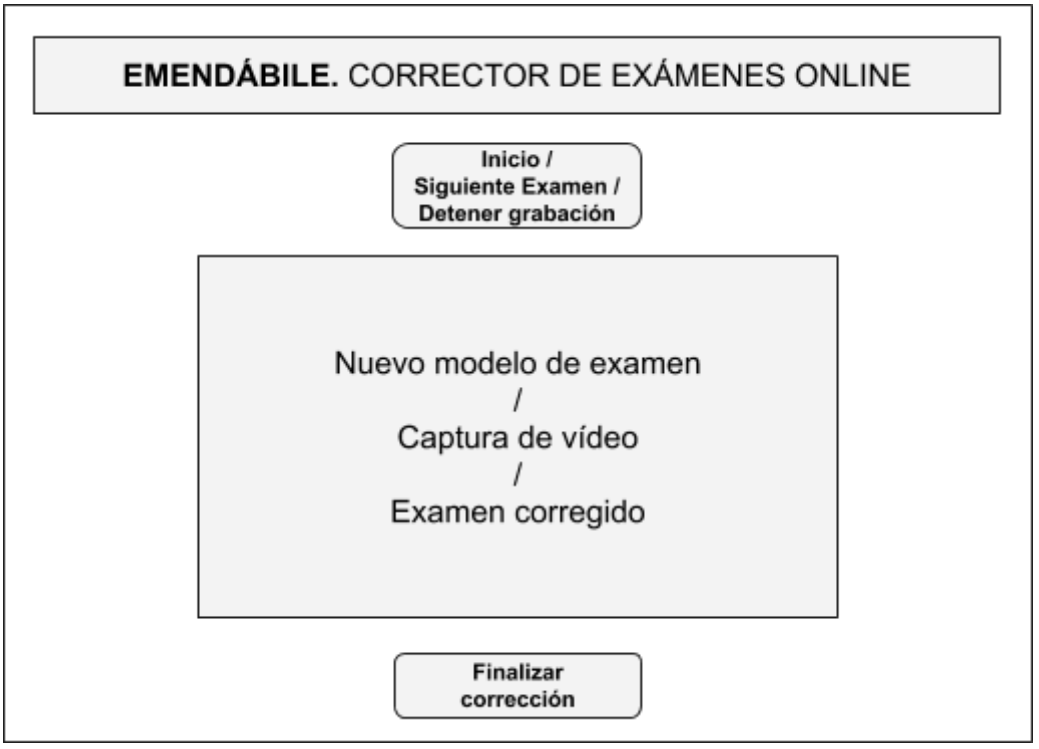

Figura 25. Estructura básica de la interfaz gráfica.

Hay cuatro bloques o estructuras funcionales principales:

1. ***Encabezado:***

Sección exclusivamente gráfica. No tiene ninguna funcionalidad aparte de la de ambientar el programa. Su implementación se hará exclusivamente con HTML y CSS.
El título del programa se marcará dentro de la sección de cabecera del documento HTML, como una sección ( ) sin nada en particular. Luego, con CSS se le dará la visualización adecuada en tamaño, fuente, contorno y alineamiento.

2. ***Inicio / siguiente examen / detener grabación:***

Botón de interacción. Deshabilitado hasta la introducción del modelo. Su función es activar o desactivar el módulo de captura.
Su implementación se desarrolla de la siguiente manera:

- En HTML se define como un botón, y se coloca en el código en un orden coherente a la estructuración global.
- En CSS se estiliza, centrándolo y añadiendo los contornos y el color adecuados.
- En JavaScript se le da la funcionalidad:
  Se implementa una función que se activa cada vez que se pincha el botón. Esta función se condiciona por el estado de captura: si se está grabando, llama al método *stopCamera*, que indica al módulo de captura que se detenga. Si no está grabando, llama al método *startCamera*, que activa la captura de vídeo.

3. ***Nuevo modelo / captura de vídeo / examen corregido:***

3.1 Nuevo modelo de examen:

Formulario de inserción de nuevo modelo (**Figuras 21 y 22**). Disponible sólo cuando se acaba de entrar en la página y ésta no dispone de datos de ningún modelo introducido. Su funcionamiento se desarrolla en comunicación con el submódulo de creación de modelo. El desarrollo es el siguiente:

- En HTML: Creación de los campos de inserción o *inputs*. Para ello se introducen varias entradas con tipo de entrada distinto (numerical, radio, etc.). Por último, se añadirá un botón para el procesamiento de los datos introducidos.

- En JavaScript: El formulario debe definir el modelo de examen, almacenado en una variable global denominada *model.* Esta variable es un objeto con los atributos: *sols (soluciones), dims (dimensiones), tipos (número de tipos), plus, deduct* y *emptiness.*
  Para ello, se comunican los datos del formulario al submódulo a partir de dos funciones que accede a los valores de cada entrada, una en cada paso.Estas funciones se se activan cuando se pincha el botón de “Siguiente” o “Iniciar corrección”.
  En el submódulo se le asignan los datos recibidos a la variable *model*.

3.2 Captura de vídeo:

Mientras la captura ha sido autorizada pero no se ha detectado aún el examen, se mostrará en este bloque la captura de vídeo que se esté llevando a cabo. Para ello, se implementa:

- En HTML: La sección en la que se mostrará la captura de la cámara consistirá en una etiqueta *video*. De este modo se indica al navegador la naturaleza del contenido.

- En JavaScript: Función definida dentro del módulo de captura. En el momento de captura del vídeo del usuario, se transmite a la estructura *"video"* del documento HTML como contenido.

3.3 Examen corregido:

En este bloque también se mostrará el resultado de la detección realizada correctamente. Además, el módulo de exposición se encargará de la rectificación del usuario ante algún posible fallo en la detección. Para esto, se debe implementar de la siguiente manera:

- En HTML: se presenta la imagen corregida como objeto imagen.

- En JavaScript:

  Resultado de la detección.

  El programa recibe del módulo de detección la matriz de decisiones, y mediante el método *draw_decisions* se encargará de dibujar en el bloque la imagen con las decisiones señaladas en verde si son correctas o en rojo si no lo son.
  Para ello, el método recibe como parámetros la imagen, las coordenadas de las celdas, el tipo de examen, la matriz de decisiones y la matriz de soluciones.

  El procedimiento es sencillo: realiza un bucle comparando la matriz de decisiones con la de soluciones (en el tipo de examen que corresponda). Si el elemento de la matriz de decisiones es un "1" y coincide con la de soluciones, se usa la función cv.Circle para pintar un círculo verde en las coordenadas de esa celda. Si es un "1" y no coincide, se pinta un círculo rojo.

  La imagen obtenida se muestra en el elemento imagen del documento HTML.

  Rectificación de corrección.

  Una vez se muestra la imagen, el programa debe ser capaz de corregir su solución en función de las interacciones del usuario. Para ello se hará uso de una función en JavaScript, llamada *check_click.*

Esta función se activa cuando se pincha en la imagen, y guarda las coordenadas donde se ha realizado el “click”.
Luego compara esas coordenadas con el conjunto de celdas:

- Si se ha marcado una celda vacía, se elimina la decisión anterior de esa fila y se añade al lugar que ocupa la celda marcada.
- Si se ha marcado una celda decidida, se elimina esa decisión.

Una vez se ha realizado la transformación, se vuelve a llamar a la función *draw_decisions* con la nueva matriz de decisiones.

4. Finalizar corrección

Último elemento. Es un botón de iteración de funcionalidad muy sencilla, su finalidad es cerrar la conexión con los módulos.

En HTML se le asignará la etiqueta de botón. Luego, con referencia a ese id, desde JavaScript se activará la función de cierre cuando se pinche sobre él.

5.2 Captura

En este módulo se definen las funciones encargadas de la comunicación con la cámara web.
Su implementación es sencilla, consta de tres funciones básicas:

1. startCamera

   Accede a la información de la webcam suministrada por el navegador:

   ```
   navigator.mediaDevices.getUserMedia({video: resolution, audio: false})
   ```

   La información obtenida es asignada al elemento video del documento HTML, dando la orden de reproducirse. Cuando el vídeo está listo para reproducirse se le añaden los atributos y se llama a la función startVideoProcessing

2. startVideoProcessing

   Es la función encargada de procesar el vídeo. Para ello define intervalos de tiempo muy breves en función del número de fotogramas por segundo de la webcam (aproximadamente 30).
   Cada vez que transcurre el intervalo de tiempo hallado captura la imagen del vídeo y la envía al módulo de detección. Si la detección es incorrecta espera otro intervalo de tiempo y repite el proceso. Si es correcta detiene la captura llamando a stopCamera.

3. stopCamera

   Pausa el vídeo, elimina los atributos añadidos en el inicio de captura del elemento vídeo HTML e informa con la variable global *streaming* que la grabación se ha detenido.

5.3 Detección

Se trata del módulo más importante de Emendábile.
Basado en Eyegrade, su función consiste en detectar el conjunto de decisiones tomadas por el alumno y el tipo de examen. Además, comparará las decisiones con las soluciones introducidas del modelo.

Siendo el diseño semejante al módulo homónimo de Eyegrade, su implementación se desarrolla de manera muy distinta. La modificación del lenguaje sobre el que se desarrolla implica una diferenciación global en la programación del código.

Para mayor ilustración del diseño e implementación del módulo, se realizará un análisis del módulo mucho más detallado, exponiendo en todo momento las diferencias más destacables.
Esta exposición no incluirá el análisis funcional de cada sección, ya que éste ya ha sido expuesto en el apartado 3.3.2

5.3.1 Preprocesado de la imagen

El preprocesamiento realizará dos filtrados a la imagen. El primero una transformación a escala de grises y el segundo un filtrado binario adaptativo.

Las funciones a utilizar de OpenCV.js son las siguientes:

- cv.cvtColor → cambio del color global de la imagen. (**Figura 26**)
- cv.adaptiveThreshold → tratamiento de cada pixel por un umbral binario adaptativo inversor. Que sea adaptativo implica que se tiene en cuenta el entorno del pixel a procesar y el contraste existente. Como es inversor los colores binarios se ven transformados a su contrario, lo negro blanco y viceversa. (**Figura 27**)

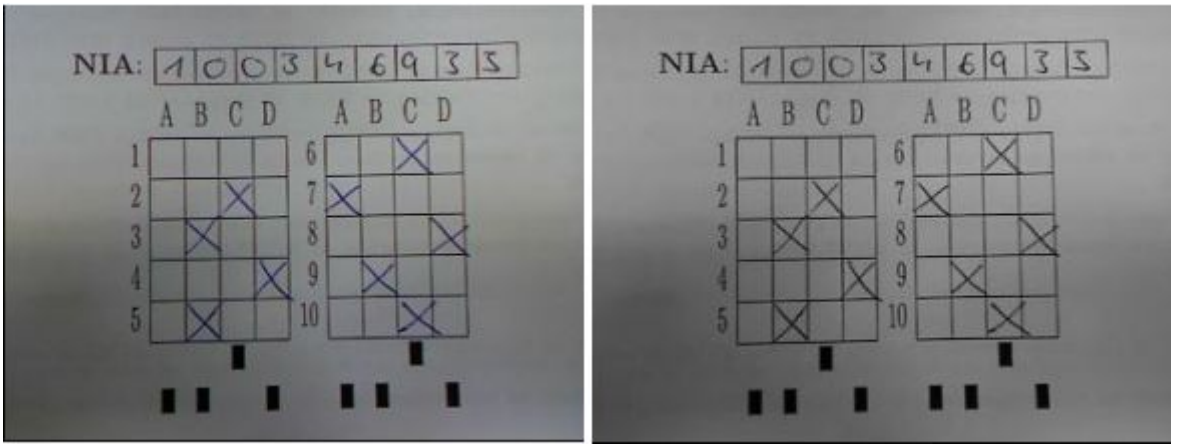

Figura 26. Transformación de imagen a escala de grises

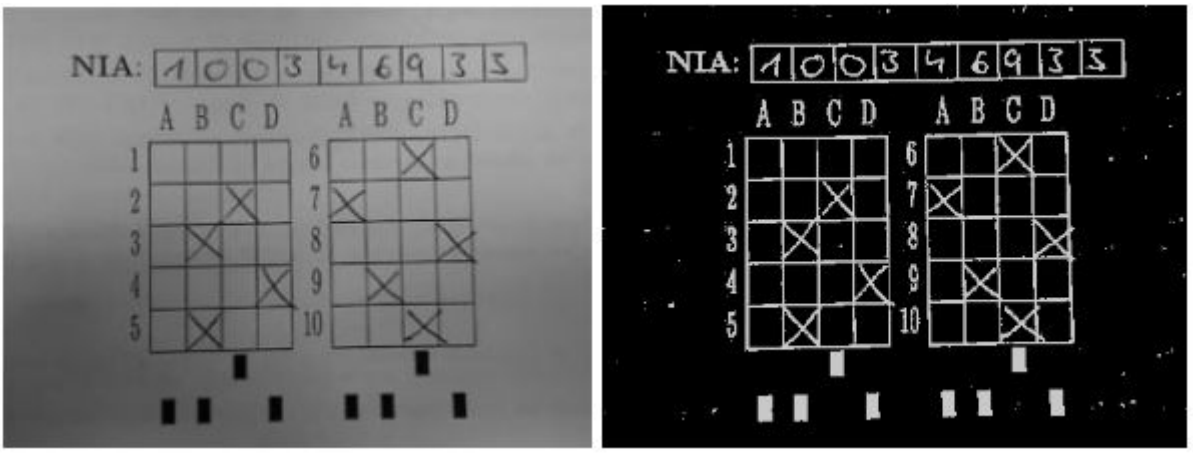

Figura 27. Transformación binaria adaptativa e inversa.

5.3.2 Detección de rectas

Una vez preprocesada la imagen ya se puede proceder con la detección de líneas. Para ello se utilizará la función cv.HoughLines.

Esta función depende sobretodo del parámetro de sensibilidad. Este parámetro a su vez no se mantiene óptimo y constante para todas las capturas, ya que depende de parámetros externos (luz, enfoque, distancia de la cámara al papel …).
A diferencia de en Eyegrade, en Emendábile se hallará su valor en función del número total de líneas detectadas.
Es decir, se inicializa un array con posibles valores del parámetro:

```
let param_hough_thresholds = [180, 160, 140, 120, 100, 80];
```

Se genera la variable *param_counter*, que gestionará la variación del parámetro de sensibilidad.
Para ello, de la captura realizada, se ejecuta la detección de líneas con el parámetro correspondiente al que indique el contador:

```
cv.HoughLines(image, lines, 1, Math.PI / 180, param_hough_thresholds[param_counter], 0,
0, 0, Math.PI);
```

Las líneas detectadas se procesarán como corresponde en la programación del código. Si la detección es correcta, el parámetro no variará. Si la detección falla en algún paso, se capturará otra imagen, que se procesará con el siguiente parámetro del array:

```
param_counter++;
if(param_counter==param_hough_thresholds.length) param_counter=0;

cv.HoughLines(image, lines, 1, Math.PI / 180, param_hough_thresholds[param_counter], 0,
0, 0, Math.PI);
```

Utilizando este método se lleva a cabo la detección de líneas de la imagen preprocesada. (**Figura 28**)

Estas líneas tienen el siguiente formato: [θ, ρ], donde θ es la inclinación y ρ la distancia al origen.

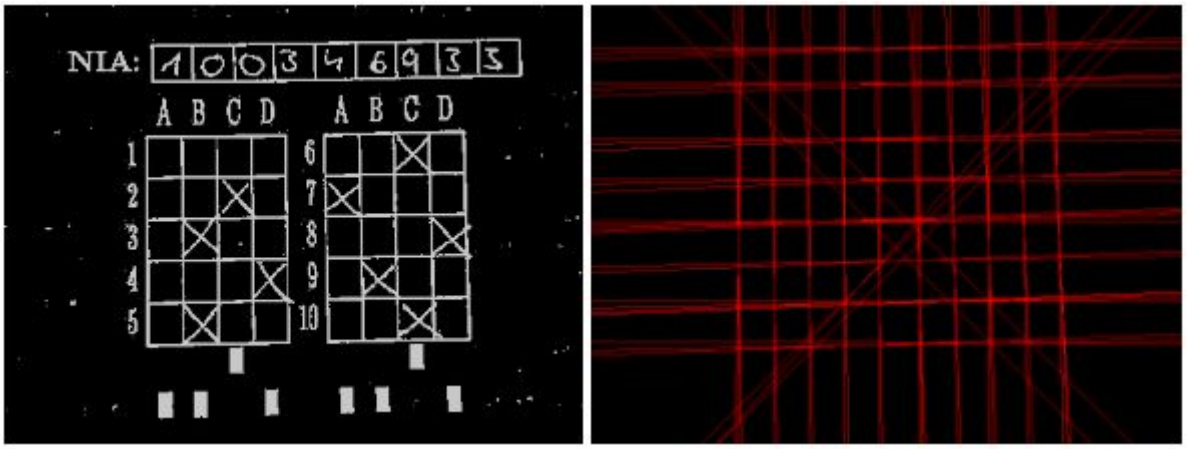

Figura 28. Detección de líneas de la imagen.

5.3.3 Detección de direcciones y ejes

Como se observa en la **Figura 28**, se han detectado líneas en múltiples direcciones: horizontales, verticales, y diagonales en ambos sentidos.
Para realizar el filtro direccional y escoger sólo las orientaciones deseadas implementamos el método *detect_boxes.*

Método detect_boxes:

Este método catalogará el conjunto total de rectas según su eje o dirección y filtrará las direcciones no deseadas. Para ello, se define el siguiente formato de eje: (θ_E, [(θ_L, ρ)]).
Donde se indica la inclinación central de dicho eje (θ_E) y el conjunto de rectas que se pueden considerar de ese eje por la aproximación de la inclinación ([(θ_L, ρ)]).

Los pasos de este etiquetado y filtrado son los siguientes:

1. Ordenación del conjunto de rectas según su inclinación.

   Para el procesamiento del método se debe ordenar el conjunto de líneas según su inclinación θ. En Eyegrade, implementado con Python, esto es sencillo y se hace en una línea de código.
   En JavaScript, sin embargo, se deberá realizar una transformación de matriz a vector (*método mat_to_array*), con una posterior ordenación mediante el método *sort*.

2. Creación de eje y comparación.

   Para generar un nuevo eje, se asigna como θ_E la θ de la primera recta que no se pueda catalogar por aproximación, según un parámetro de sensibilidad.
   De este modo, se recorre el total de rectas, introduciendo éstas en los ejes existentes, y si no se puede, generando un nuevo eje.

3. Reorganización de ejes.

   - Se fusionan los ejes cuya inclinación dista π radianes.

   - Se establece θ_E como la media de todas las θ_L

4. Filtrado de ejes no deseados

   Lo primero, en función de las dimensiones del modelo, se rechaza la captura si el conjunto de líneas no cuenta como mínimo con el número de líneas necesarias para la formación de tabla de respuestas.

   Si se encuentran suficientes líneas se filtrará el número de ejes hasta que sólo haya dos: horizontal y vertical.

   Para ello, se seleccionará sólo el par de ejes que sean perpendiculares entre sí, llamando a la función *angles_perpendicular.* Si hay dos pares de ejes perpendiculares, se seleccionarán aquellos cuya inclinación sea más cercana a π y a π/2.

El resultado total del filtrado de direcciones es el siguiente: (**Figura 29**)

Figura 29. Filtrado de direcciones

5.3.4 Filtrado de rectas

Para la correcta detección de esquinas será necesario el procesamiento de una línea virtual por cada línea real, luego habrá que filtrar el conjunto de líneas. Para ello se llama a la función *filter_axes*.

Filter_axes procesa las rectas según su orientación. Primero las horizontales y luego las verticales.
Una vez se han dividido por orientación, la función de filtro realiza los siguientes pasos:

1. Se ordena el conjunto de líneas en función de la distancia al origen (ρ), teniendo en cuenta que todas las líneas tienen aproximadamente la misma inclinación.
2. Se selecciona la primera recta como referencia y se crea el array *lines*, donde se almacenarán las rectas resultantes del filtro.
3. Se recorre el conjunto de líneas, comparándose con la referencia.
    - Si la diferencia de distancias es menor que cierto umbral se coloca la nueva línea como referencia.
    - Si la diferencia de distancias es mayor que el umbral, se añade al array *lines* una nueva línea formada por la media de los atributos θ y ρ de todas las líneas anteriores que no habían superado el umbral. Luego se pone la nueva recta como referencia.

En el array *lines* se encuentran las líneas filtradas. Si había varias líneas cercanas, se han sustituido por una única línea de orientación y distancia medias.

El resultado visual del filtrado es el siguiente: (**Figura 30**)

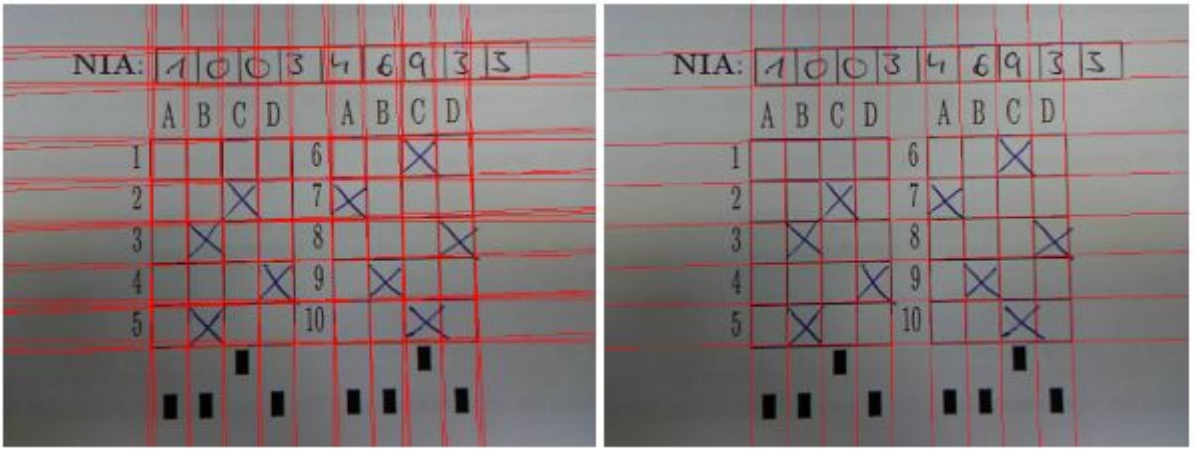

Figura 30. Filtrado de rectas en función de su separación.

5.3.5 Detección de celdas

Teniendo pues almacenado el número exacto de líneas que definen toda la cuadrícula, ya se puede proceder con la detección de celdas.

Cada celda se define por las cuatro esquinas que la forman. Por ello, se formará primero la matriz de esquinas de la imagen, hallando las intersecciones de las rectas.

Para el ejemplo tomado, la matriz de esquinas debe ser de dimensiones (2 x 6 x 5), ya que se divide la matriz en dos submatrices, una para cada subtabla de la imagen.

Para formar la matriz de esquinas, se aplica el método *cell_corners*. Este método realiza los siguientes procesos:

1. Reordena el array de líneas horizontales comenzando por abajo. De este modo no se tienen en cuenta para hallar el conjunto de esquinas las rectas del área de identificación o sobrantes superiores.

2. En función de las dimensiones, se hallan las líneas esperadas en cada dirección, llamando a la función *expected_lines* (**Figura 31**). Si el número hallado es distinto del detectado se desecha la captura.

```
function expected_lines(dimensions){
    let expected = {h:0, v:0};
    let c=0;
    let c_max=0;
    for(let i=0;i<dimensions.length;i++){c+=dimensions[i][0];}
    expected.v = dimensions.length + c;
    for(let i=0;i<dimensions.length;i++){
        c=dimensions[i][1];
        if(c>c_max){c_max=c;}
    }
    expected.h = 1 + c_max;
    return expected;
}
```

Figura 31. Código para la comprobación del número de líneas.

3. Se realiza un bucle de intersección entre todas las líneas horizontales y verticales, llamando a la función *intersection*.

4. Cada resultado se almacena en la matriz de esquinas.

Al contrario que en Eyegrade, Emendábile no utiliza duplas como formato de los puntos, sino que los instancia como objetos con las coordenadas como atributos. Esto mejora y facilita el desarrollo y transformación de los puntos.

Para nuestro ejemplo, obtenemos una matriz de (2 x 6 x 5), de elementos punto, con un formato:

```
punto = {x: coorX, y: coorY}; → punto.x = coorX; punto.y =
                                coorY;
```

La representación visual de estos puntos es la siguiente: (**Figura 32**)

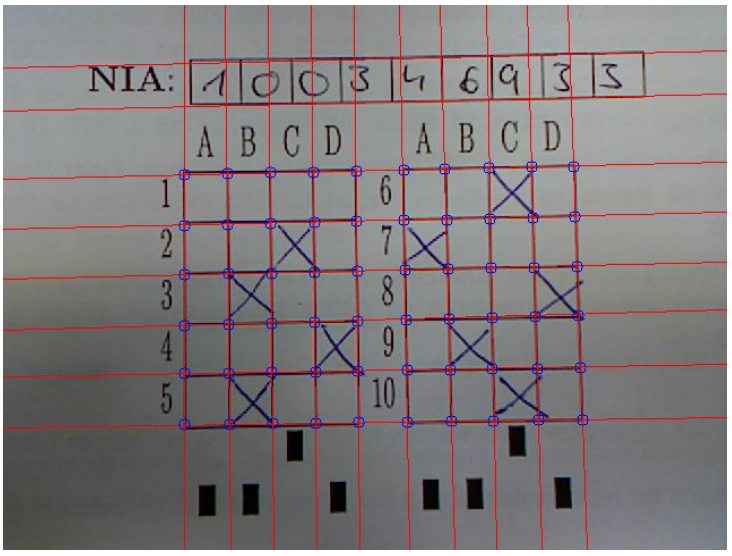

Figura 32. Puntos hallados por la intersección de las líneas

5.3.6 Detección de respuestas

Una vez establecida la localización de cada casilla, se podrá decidir si ésta está marcada o no. Para ello se hará uso de la función *decide_cells,* que recibiendo la imagen preprocesada y la matriz de esquinas, procede de la siguiente forma:

1. Se halla la matriz de celdas.

   Llamando al método *answer_cells_geometry* se genera la matriz de celdas. Cada una de ellas definida por su cuatro esquinas (plu → point left up, prd → point right down, etc.).

   El método es sencillo, recorre el conjunto de esquinas asignando las cuatro correspondientes a cada celda: (**Figura 33**)

```
function answer_cells_geometry(corners){
    let cells = new Array();
    for(let i = 0; i < corners.length; ++i){
        for(let j = 0; j < corners[i].length-1; j++) {
            let row = new Array();
            for(let k = 0; k < corners[i][j].length-1; k++){
                let cell = {plu: corners[i][j][k], pru: corners[i][j][k+1],
                            pld: corners[i][j+1][k], prd: corners[i][j+1][k+1]};
                row.push(cell);
            }
            cells.push(row);
        }
    }
    return cells;
}
```

Figura 33. Creación de matriz de celdas.

2. Se decide cada celda.

   Se recorre en bucle cada una de las celdas, aplicando la función *is_cross*, que devuelve un '1' si la celda está marcada y un '0' si no.

   La función *is_cross* está implementada de forma distinta a Eyegrade, tal y como se expuso en el apartado 3.3.2.
   Al no poder manejar máquinas SVM mediante la librería OpenCV.js, la lectura de cruces por esta vía es imposible, por lo que el método variará y se basará en el mecanismo de detección de *infobits* de Eyegrade.

   En detalle, la función *is_cross* realiza los siguientes pasos:

   a. Generación de la máscara de comparación.
      Para la celda correspondiente (**Figuras 34 y 37**) se genera una máscara de cruz con la que se marcaría la casilla (**Figuras 35 y 38**).

   b. Multiplicación pixel a pixel de la imagen preprocesada con la máscara.
      El resultado de la multiplicación se guarda en la imagen *producto* (**Figuras 36 y 39**)

   c. Análisis del resultado.
      Si el producto contiene cierto número de píxeles activos (mayor que cierto parámetro de sensibilidad), se considera que la casilla está marcada. Si no, la casilla está vacía.

      Como se puede observar en el ejemplo de las **Figuras 36 y 39**, el resultado de la multiplicación muestra claramente si la casilla está marcada o no.

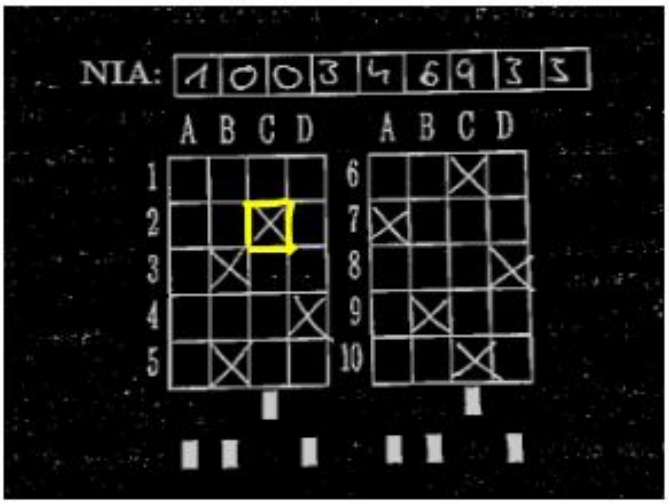

Figura 34. Casilla marcada

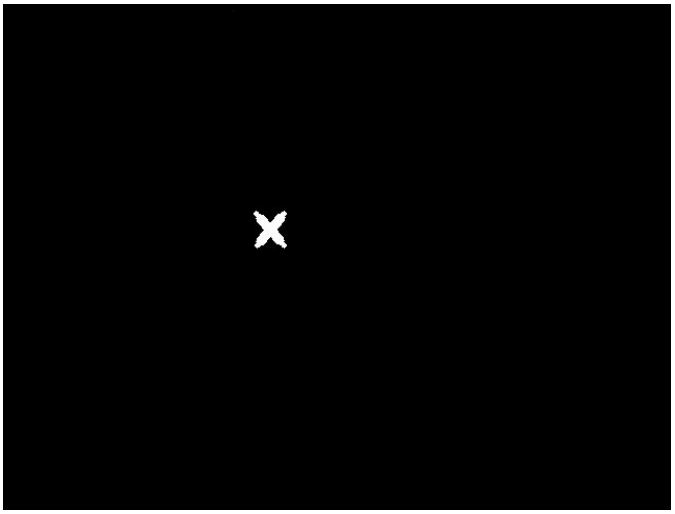

Figura 35. Máscara de casilla marcada

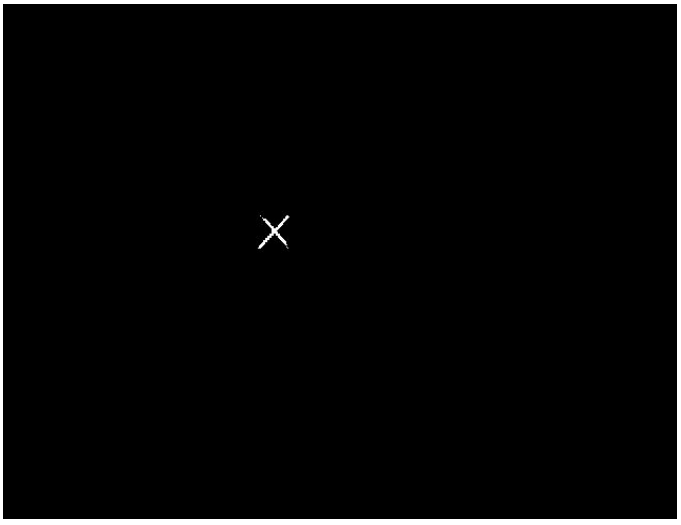

Figura 36. Resultado de la multiplicación. Casilla marcada.

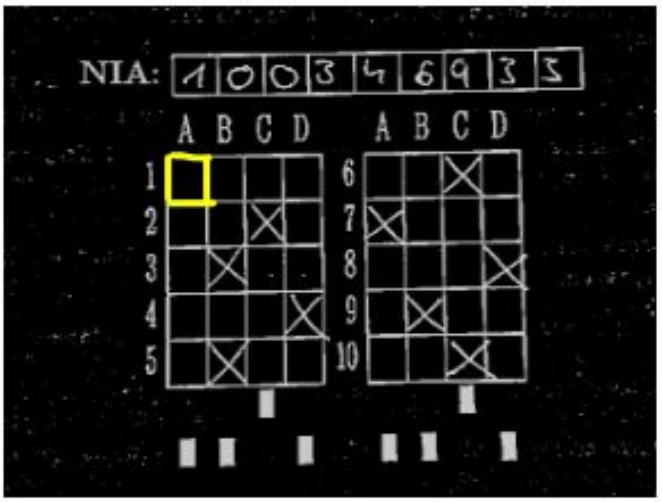

Figura 37. Casilla no marcada

Figura 38. Máscara de casilla no marcada

Figura 39. Resultado de la multiplicación. Casilla no marcada.

3. Se obtiene la matriz de decisiones.

   Una vez conocido el estado de cada celda, se introducen estos datos en una matriz con el formato equivalente al del siguiente ejemplo: **(Figura 40**)

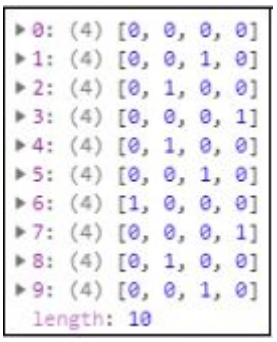

Figura 40. Ejemplo de matriz de decisiones.

De este modo ya conocemos el conjunto de decisiones tomadas por el alumno.

5.3.7 Detección de tipo de examen

Para la detección del tipo de examen se aplica el mismo procedimiento que para la detección de cruces.

Se selecciona el área en el que se va a buscar el infobit. Que es equivalente a una (‘1’) o dos (‘0’) celdas por debajo de la última fila.

Para ello, se genera una máscara circular en el área correspondiente (**Figura 42**). Y, del mismo modo que con la detección de cruces, se multiplica pixel a pixel por la imagen preprocesada (**Figura 41**). Si el resultado (**Figura 43**) cuenta con un número de píxeles positivos mayor que cierto umbral, entonces se ha detectado un infobit en ese área.

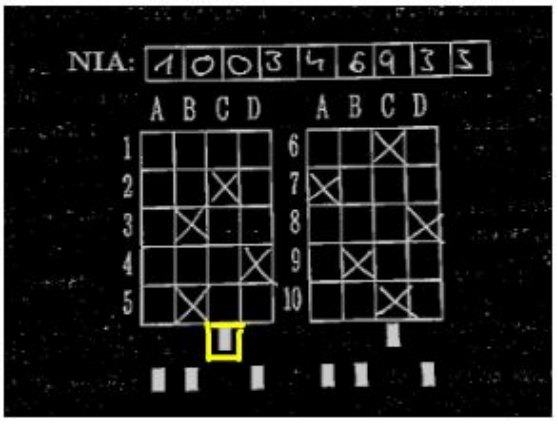

Figura 41. Infobit positivo en imagen preprocesada.

Figura 42. Máscara circular de infobit positivo.

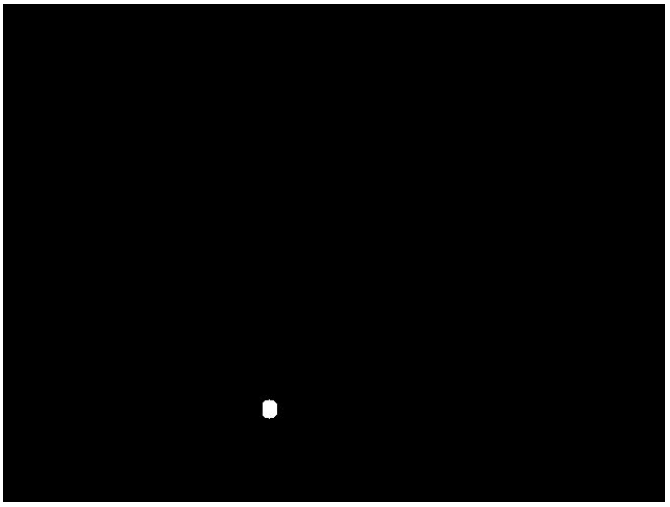

Figura 43. Resultado de la multiplicación. Infobit positivo detectado

Si no se detecta un infobit en el área superior, se ha de detectar en el inferior. Si no es así, se desecha la captura, pues ésta no incluye el conjunto completo de infobits.

5.3.8 Corrección de decisiones

Para la asignación de notas, se compara la matriz de soluciones del modelo con la de decisiones de la detección.
Para ello se llama al método *correction*, que recibe como parámetro el modelo, el tipo de examen detectado y la matriz de decisiones.

Se selecciona de la matriz solutions la submatriz correspondiente al tipo hallado. Una vez se tienen las dos matrices *sols_tipo* y *decisiones*, equivalentes en dimensiones, comienza la comparación.

Para ello se recorren las posiciones de ambas matrices. En cada casilla:

- Si está marcada (el elemento de la matriz de decisiones es '1'), la variable *row_marked* se pone a "*true".* Si la solución en esa posición es '1' se incrementa la variable *correct,* si no, se incrementa la variable *incorrect.*

- Si se completa la fila y no ha sido marcada, es decir, la variable *row_marked* está en *"false",* se incrementa la variable *empty*.

La nota final se calcula a partir del puntaje introducido y el número de respuestas de cada tipo (correcta, incorrecta, vacía). Para almacenarse se genera un nuevo objeto *puntuation*, que almacena como atributos la nota final, y el número de respuestas correctas, incorrectas y vacías.
La implementación del código se puede ver con mayor claridad en la **Figura 44**:

```
function correction(sols, decisions, type, plus, deduct, emptiness){
    let correct=0;
    let incorrect=0;
    let empty=0;
    let row_marked=false;
    let solutions = sols[type];
    for (let i = 0; i < decisions.length; i++) {
        for (let j = 0; j < decisions[i].length; j++) {
            if(decisions[i][j]==1){
                row_marked=true;
                if(solutions[i][j]==1){
                    correct++;
                }else{
                    incorrect++;
                }
            }
        }
        if(!row_marked){
            empty++;
            row_marked=false;
        }
    }
    let puntuation = {note:(plus*correct + deduct*incorrect + emptiness*empty),
                      correct: correct, incorrect:incorrect, empty:empty};

    return puntuation;
}
```

Figura 44. Código de la función *correction,* para la asignación de nota.

5.3.9 Función process

Todas estas acciones se llevan a cabo desde la función *process*, que es a la que se accede desde el módulo de captura.

Esta función hace un llamamiento secuencial de los métodos necesarios para la corrección. Si algún método indica que la captura no es válida, *process* informa al módulo de captura, que procede de la misma forma con la siguiente imagen.

Con esta exposición se completa el desarrollo y diseño del módulo de detección.

Como se ha explicado en apartados anteriores, no ha sido posible implementar la funcionalidad de reconocimiento de identificador del alumno, debido a la falta de soporte provisto por OpenCV.js actualmente. Esto impide almacenar las notas específicas para cada alumno y su posterior exportación.

Sin embargo, se considera que el desarrollo realizado es suficiente para un uso adecuado y fructífero de la aplicación.

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1 Conclusiones

Finalizado el desarrollo del proyecto, se compararán los resultados obtenidos con los objetivos iniciales. Además, se expondrán las dificultades de mayor carácter que han tenido lugar en el desarrollo global de Emendábile.

El conjunto de los objetivos planteados para el desarrollo global es el siguiente:

1. Desarrollo de la aplicación en entorno Web, con acceso libre desde el navegador.
2. Inserción del modelo de examen.
3. Detección del área de soluciones, tipo de examen y respuestas dadas mediante una cámara digital.
4. Muestra de corrección, incluyendo nota final y calidad de cada respuesta.
5. Rectificación de corrección.
6. Interfaz gráfica sencilla e intuitiva.
7. Distribución del programa mediante licencia de código abierto.

Como se ha expuesto en los capítulos 4 y 5, todas estas funcionalidades han sido implementadas correctamente en el proyecto, por lo que se considera que el programa resultante es un prototipo de Emendábile completo y eficaz.

Las principales dificultades salvadas en la realización del proyecto han sido las siguientes:

- *Optimización en el uso del lenguaje JavaScript:*

  Tratándose de la herramienta más utilizada en el desarrollo del proyecto, se ha necesitado de mucho tiempo de aprendizaje y desarrollo para la óptima implementación del programa, ya que dicho lenguaje de programación nunca había sido usado por el desarrollador.

- *Comunicación con la webcam:*

  Al contrario que en Eyegrade, Emendábile accede a los datos de la cámara web a través del navegador. Esto ha supuesto dificultades en la implementación por la falta de precedentes y dificultad que entraña la selección entre los dispositivos posibles de captura de vídeo desde el navegador.

- *Implementación en el servidor:*

  La puesta en marcha del servicio se inició en un servidor propio, accesible directamente por direccionamiento ip. Esto derivó en un problema, y es que el programa no se servía bajo https seguro, sino sólo bajo http.
  Esto provocó que el navegador del cliente no considere segura la página y no admita la transmisión de contenido multimedia, por lo que la herramienta pierde su utilidad por completo.

  Para solucionarlo, se ha utilizado el servicio gratuito de *GitHub Pages* [25], para ofrecer el servicio bajo un nuevo dominio y en uso del protocolo https.

Además de las dificultades mencionadas, se han de añadir dificultades en las implementaciones de funcionalidades de Eyegrade, debido al cambio de lenguaje y entorno. (Formulario de inserción de examen, rectificación de corrección, tratamiento de matrices y *arrays*, etc.)

6.2 Trabajos futuros

Queda demostrado entonces, por la realización del proyecto, que es viable realizar una reimplementación de Eyegrade en un entorno web, adecuando sus funcionalidades principales, a excepción de la detección mediante máquinas SVM del identificador del alumno.

Teniendo en cuenta que toda la reimplementación de Eyegrade es un trabajo de magnitud desproporcionada al entorno de realización del proyecto, sí se podría continuar el desarrollo de Emendábile fuera de este ámbito añadiendo el conjunto de funcionalidades de Eyegrade aún no introducidas.

Por ello, se planteará como futuro trabajo la amplificación de Emendábilis a partir de las siguientes funcionalidades:

- Importación de ficheros de configuración. Estos ahorrarán tiempo de uso dando la posibilidad de incluir directamente los atributos del modelo de examen o de introducir el listado de alumnos examinados.

- Exportación del listado de notas. Permitir la descarga de un fichero que contenga el identificador o nombre de cada alumno con su nota resultante.

- Estudio y posible implementación del reconocimiento de dígitos. Estudiar, analizar e implementar posibles variantes que permitan al programa la detección del estudiante y su posterior asignación de nota automática.

7. MARCO REGULADOR

Para la correcta implementación del proyecto, se ha debido tener en cuenta el entorno regulador en el que se ha desarrollado. Dentro de este marco se engloban todas las legislaciones que pueden afectar a su desarrollo, mantenimiento y distribución.

Los casos de análisis totales son los siguientes:

7.1 Reglamento general de protección de datos

El reglamento RGPD [26] regula la legislación referente a la manipulación, conservación y divulgación de material personal y privado.

En este proyecto se ha desarrollado el primer prototipo de Emendábile. Este programa busca la corrección de exámenes online, y tiene como algunos de sus objetivos implementar funciones existentes en Eyegrade (*Sección 1.2*).

Sin embargo, no consta como objetivo del proyecto implementar la función de inserción de alumnos y exportación del listado de notas, vigente en Eyegrade, por lo que no se va a ver afectado de ninguna forma por esta regulación.

Además, si dicha funcionalidad se implementase en trabajos posteriores, no supondría ningún problema. Ya que, al tratarse de procesamiento desde el lado del cliente, el servidor no recibiría ni almacenaría en ningún momento datos del carácter personal de los alumnos.

Toda la gestión de estos datos y el correcto seguimiento del reglamento RGPD serían responsabilidad total del usuario, ya que sería el único en el uso de la aplicación con acceso a estos datos de carácter privado.

7.2 Licencias de herramientas utilizadas

1. ***OpenCV:***

   La librería utilizada como herramienta de visión artificial cuenta con una licencia BSD, que nos permite sin inconveniente alguno el uso y distribución realizados por Emendábile.

2. ***Eyegrade:***

   Muchas de las funciones del programa están basadas en implementaciones de Eyegrade, por lo que se trata de la referencia principal de Emendábile.

   Sin embargo, esto no supone ningún inconveniente legislativo, ya que Eyegrade cuenta con licencia GNU General Public License (GPL), versión 3 o superior [27]. Esto nos permite el uso referenciado de contenido y la posterior divulgación del producto resultante.

7.3 Licencia propia de Emendábile

Emendábile usará, por motivos éticos, sociales y de mayor divulgación, la licencia GNU General Public License (GPL), que permite la referenciación y utilización adecuadas del código y el uso gratuito e ilimitado del programa.

8. ENTORNO SOCIO-ECONÓMICO

En el entorno de su desarrollo, el proyecto realizado generará un principal beneficio de carácter socio-económico, el ahorro de recursos de los docentes.

El objetivo principal del proyecto es aumentar la eficiencia de las correcciones de exámenes tipo test, reduciendo de este modo el tiempo empleado por los docentes para dichas tareas.

Esto generará ventajas sociales y económicas en el entorno del docente que le dé uso, ya que tendrá más recursos temporales que podrá emplear en una mejor docencia (preparación de clases, atención a los alumnos, etc.).

LISTA DE ACRÓNIMOS

***BSD**. Berkeley Software Distribution*

***CSS**. Cascading Style Sheets*

***CV**. Computer Vision*

***GNU**. GNU Not Unix.*

***HTML**. HyperText Markup Language*

***HTTP**. HyperText Transfer Protocol*

***IOS**. Iphone Operating System*

***JS**. JavaScript*

***PC**. Personal Computer*

***PHP**. Hypertext Preprocessor*

***RGPD**. Reglamento General de Protección de Datos*

***SVM**. Support Vector Machine*

***WWW**. World Wide Web*

REFERENCIAS BIBLIOGRÁFICAS

[1] J. A. Fisteus, “grading multiple choice exams with a webcam,” *Eyegrade*. [Online]. Available: http://www.eyegrade.org/. 29-Mar-2019

[2] “How Does Client-Side Scripting Improve Web Application Performance?,” *AnAr Solutions Pvt. Ltd*. [Online]. Available: http://www.anarsolutions.com/client-side-scripting/. 19-May-2019

[3] R. Camden, *Client-side data storage keep it local*. O'Reilly, 2016.

[4] A. Ullah, “1: Client-side Scripting Features,” *Differences between Client-side and Server-side Scripting*. [Online]. Available: https://www.sqa.org.uk/e-learning/ClientSide01CD/page_18.htm. 29-May-2019

[5] C. Wakefield, “Chapter 10 - Developing Web Applications”, en *VB.NET developer's guide*. Syngress Media, 2001, pp.459-522.

[6] D. Herron, *Node Web development a practical introduction to Node, the exciting new server-side JavaScript web development stack*. Birmingham, U.K.: Packt Pub., 2011.

[7] “World Wide Web Consortium,” *Wikipedia*. [Online]. Available: https://es.wikipedia.org/wiki/World_Wide_Web_Consortium. 03-Jun-2019

[8] González Díaz, David, Universitat Autònoma De Barcelona. Escola D&apos, Enginyeria, and Meneses Benítez, Montserrat, “Desarrollo y gestión de una página web educativa: Wolframio, un tipo con química,” 2012.

[9] “Lista de Elementos HTML5,” *Documentación web de MDN*. [Online]. Available: https://developer.mozilla.org/es/docs/HTML/HTML5/HTML5_lista_elementos. 03-Jun-2019

[10] C. Casciano, *CSS*. Peachpit Press, 2010.

[11] Powell, Thomas A and Schneider, Fritz, *JavaScript*. 3rd ed, McGraw-Hill, 2012.

[12] “What is JavaScript?,” *Documentación web de MDN*. [Online]. Available: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qué_es_JavaScript 29-May-2019

[13] G. R. Bradski and A. Kaehler, “Chapter 1: Overview”, en *Learning OpenCV : computer vision with the OpenCV Library*, First edition, O'Reilly, 2008.

[14] “Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN,YOLO,SSD,” *CV*. [Online]. Available: https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/. 02-Jun-2019

[15] “Converting Image Formats - Learn ImageJ,” *Google Sites*. [Online]. Available:https://sites.google.com/site/learnimagej/tutorials/converting-image-formats 03-Jun-2019

[16] “Biblioteca (informática),” *Wikipedia*. [Online]. Available: https://es.wikipedia.org/wiki/Biblioteca_(informática). 03-Jun-2019

[17] “OpenCV,” Licence *OpenCV*. [Online]. Available: https://opencv.org/license/ 03-Jun-2019

[18] “Introduction to OpenCV.js and Tutorials,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4/df/d0a/tutorial_js_intro.html. 03-Jun-2019

[19] “Hough Transform in OpenCV,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4/d3/de6/tutorial_js_houghlines.html. 03-Jun-2019

[20] Gilday, “gilday/fast-grades,” *GitHub*. [Online]. Available: https://github.com/gilday/fast-grades. 03-Jun-2019

[21] Z. G. LLC, “iPhone and Android Grading App for formative assessment and quizzes.,” *ZipGrade*. [Online]. Available: https://www.zipgrade.com/. 05-Jun-2019

[22] “AutomateTest GradingwithGrade Your Test Application,” *GradeYourTest.com*. [Online]. Available: https://www.gradeyourtest.com/. 05-Jun-2019

[23] “Exam Reader - exam / test grading app for mobile phones & tablets,” *bebyaz yazilim*. [Online]. Available: https://bebyaz.com/ExamReader. 05-Jun-2019

[24] “Online Grader and Grading App for Teachers,” *GradeCam*. [Online]. Available: https://gradecam.com/. 05-Jun-2019

[25] “GitHub Pages,” *GitHub Pages*. [Online]. Available: https://pages.github.com/. 07-Jun-2019

[26] “Nuevo Reglamento - Introducción,” *RGPD*. [Online]. Available: https://rgpd.es/. 07-Jun-2019

[27] “gnu.org,” *[A GNU head]*. [Online]. Available: http://www.gnu.org/licenses/gpl-3.0.html. 07-Jun-2019

ANEXOS

ANEXO A: ABSTRACT

1. Introduction

The following abstract will briefly exhibit the analysis and development done to implement Emendábile, a multiple choice exams grader.

Emendábile is an online application, free to use, that detects and corrects test type exams with a specific format using a digital camera.

1.1 Motivations

The main motivation of the project has been the possibility of producing a qualitative improvement in the grading process of teachers, thus saving their own resources and therefore contributing to a more efficient education.

To achieve this goal, a free, immediate, intuitive, and low-cost application (without the need for scanners or other expensive tools) will be required.

To bring these qualities to their fullest, the project will be developed in a Web environment, using JavaScript. This will give the program a high differentiability compared to similar tools.

Eyegrade, a desktop application with an open source license, will be established as the main reference in the development of the program, due to its similar functions.

1.2 Objectives

The main objective of the project will be the Web implementation of an application of automatic grading of multiple choice exams, making use of Computer Vision tools on the images captured by a digital camera.

For this, the following objectives will be defined:

- Insertion of the examination model, making possible the correction to the program.
- Detection of the exam. Grid processing, type of exam and decisions.
- Grading representation. Display of the correction made.
- Rectification of correction. Allows correction of program errors.
- Intuitive user interface.
- Free program, distributed with open source license.

2. Developmental environment

We will now expose the reference models in the design and the tools used for the implementation.

2.1 Web architecture

For the development of the Web architecture, two models are established as viable alternatives: Client-Side development and Server-Side development.

The Client-Side model establishes the execution environment in the client, that is, the user's browser is responsible for the compilation and execution of the code.

The main advantages of this model are the savings in execution infrastructures (central servers), increased privacy and shorter code execution time with low computational cost.

The Server-Side model establishes the server as the main execution environment of the program. That is, the user transmits the initialization data to the server, which executes the code and retransmits the result of the operation.

The main advantages of this model is the increase in security, automatic content update and shorter code execution time with high computational cost.

In accordance with the objectives of the project, the use of the Client-Side model has been decided for its implementation, due to the prioritization of its advantages. JavaScript, HTML and CSS languages will be used as the main implementation tools.

2.2 Computer Vision

The main tool that has made the artificial vision functionalities possible has been the OpenCV library.

This library has been used for the following reasons:

- It is developed on an open source license, which allows unlimited and free use of the tool.

- It has a recently implemented module: OpenCv.js, which allows and adapts the use of this library to the JavaScript language, the main tool of this project.

2.3 Eyegrade

Eyegrade is the Emendábile's main reference program.

The operation of both programs is very similar, but the main difference is that Eyegrade is not implemented as an online tool, but as a desktop application. This will entail differences in the whole development of the implementation (programming language, user interface, execution environment, communication with the client, etc.)

Due to the similarity of functionalities and the open-source license of Eyegrade, this program has been established as a conceptual reference in some of the main functionalities of the project.

3. Design and implementation

The implementation of the program has been developed according to a modular architecture. That is, a set of main modules with specific functions have been defined, and they have been developed independently, generating in addition the set of functions necessary for their communication.

The architecture of the program consists of the following operation modules:

Exposition

This module manages the interaction with the user and the graphic layout of the program.
Its implementation is based on HTML, CSS and JavaScript. The main functionalities included are the following:

- Graphic layout. Through HTML and CSS it generates a simple and intuitive visual interface.

- Insertion of the new model. It works in communication with the model insert submodule. It manages the dynamic forms for entering exam data (types, dimensions, solutions and score).

- New exam, stop correction. The exposition module communicates the decisions made by the user to the rest of the modules, based on the interaction development.

- Solutions representation. The module represents to the user the result of the processing of the exam: correct and incorrect answers and final grade.

- Grading correction. It allows the user to correct the solution obtained if he does not consider it correct.

- Ending. It gives the user the possibility of completing the total correction, closing his session.

Capture

The capture module manages the communication with the digital camera. Its function is to capture the image of the video that is received. It is executed when it is indicated by the exposure module by the user's indication, or when the detection of the exam has not occurred correctly and a new image is required.

Detection

It is the most important module of the program. Its function is to detect the set of decisions made by the student. For this, it will use the OpenCV computer vision library and the JavaScript programming language.

So that the program can detect and correct the exam, it must be in the proper format. This format consists of the table of final answers, indicated with a cross each box corresponding to the marked option.

The set of main features included in this module are the following:

- Preprocessing of the image. The image must be transformed for a better detection of its content. This transformation will consist in a maximum increase of the contrast through binary and inverse filtering of each pixel (or white, or black).

- Detection of lines. From the functions implemented in OpenCV, the real lines existing in the image will be detected.

- Address detection. From all the lines the addresses will be filtered, resulting lines only of to two possible axes: vertical and horizontal.

- Filtering of lines. Of all the set of lines, all those that do not have a real representation in the image will be filtered.

- Cell detection. Once the set of real lines is detected, the points of intersection will be found, which will define the response cells.

- Answers detection. Within each of these response cells, the cross detection method will be executed. Which, by comparison, decides whether the box is checked or not.

- Detection of the exam type. From the infobits, present in the lower part of the exam, the type of exam to be corrected is detected.

- Decisions evaluation. Once the set of decisions has been detected, and according to the decided type, the final correction and the assignment of the exam grade are done by comparison with the model introduced.

4. Conclusions and future work.

The result developed, a program that meets all the initial objectives, can be considered a complete and effective prototype of Emendábile.

The implemented program is now a Web application, capable of detecting and grading exams based on computer vision, with an intuitive user interface and a simple and free use.

For these reasons it is established that Emendábile is a complete and suitable tool for its use.

However, due to the lack of temporary resources, the reimplementation of all the functionality of Eyegrade in Emendábile is not possible, so the implementation of the following functionalities is proposed as future work streams:

- Detection of the student's identifier. This will automate the assignment of grades to the list of students entered.

- Configuration files import. It will simplify the insertion of models and list of students from the import and reading of text files.

- Export of evaluation lists. It will allow the user to keep the corrections (students and their respective grades) in a downloadable format external to Emendábile.